PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-216043

(43)Date of publication of application : 10.08.2001

(51)Int.Cl. G06F 1/00
G09C 1/00

(21)Application number : 2000-354265

(22)Date of filing : 21.11.2000

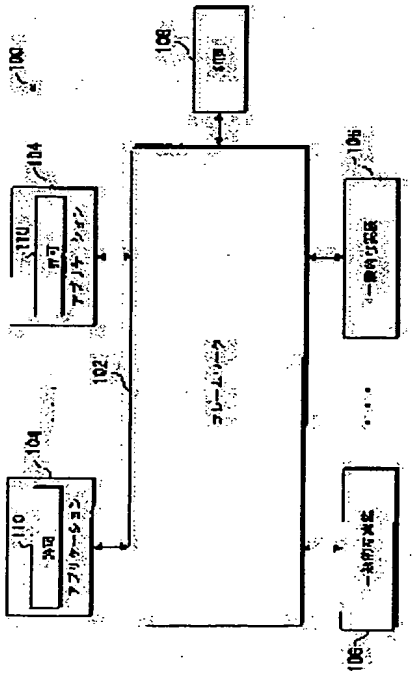
(71)Applicant : SUN MICROSYST INC

(72)Inventor : LIU SHARON S
LUEHE JAN

(30)Priority

Priority number :	Priority date :	Priority country :
1999 166871	22.11.1999	US
2000 174630	05.01.2000	US
2000 483246	14.01.2000	US

(54) MECHANISM FOR DECIDING CONSTRAINT TO BE CHARGED ON MOUNTING OF SERVICE



(57)Abstract:

PROBLEM TO BE SOLVED: To allow a framework to certainly charge necessary constraint on a service to be offered to an application by the dynamic construction of required mounting.

SOLUTION: An application issues a request for mounting for a specific service to a framework. The framework receives this request, and when any constraint is present corresponding to this request, the framework decides which constraint should be charged on the requested mounting. This constraint is decided by judging whether or not the application issuing the request for mounting is given authorization, and when this authorization is

present, this authorization is processed for introducing the set of constraints to be charged on the mounting. This authorization is processed so that the introduced set of constraints can be the minimum constraint level.

CLAIMS

[Claim(s)]

[Claim 1]. It can set to a system as which mounting of specific service is required by application. Are the method of determining restrictions imposed on said mounting, and it is judged whether there is any permission given to said application which requires said mounting, A method provided with processing said permission in order to lead a set of restrictions imposed on said mounting according to judgment that there is at least one permission given to said application.

[Claim 2]A method according to claim 1 of being what is processed so that said permission may serve as a restrictions degree with the smallest set of said restrictions.

[Claim 3]A method according to claim 1 further provided with accessing a set of initial restriction according to judgment that there is no permission given to said application, and drawing said restrictions based on said initial restriction.

[Claim 4]said initial restriction -- two or more law -- a method according to claim 3 of being what is drawn by merging a policy and extracting restrictions restrictions from there.

[Claim 5]A method according to claim 1 provided with preparing an index of said processing of said permission being what does not have restriction of said mounting according to judgment that said permission judges whether it is the permission in which all are included, and said permission is the permission which includes all.

[Claim 6]A method according to claim 1 for which it had leading a set of said restrictions based on said permission according to judgment that said permission judges whether said processing of said permission needs to mount an exemption mechanism and said permission does not need to mount an exemption mechanism.

[Claim 7]A method according to claim 6 provided with leading a set of said restrictions based on said parameter according to judgment that drawing said restrictions judged whether said permission would have specified a set of a parameter and said permission has specified a set of a parameter.

[Claim 8]A method according to claim 6 provided with preparing an index of being a thing without restriction of said mounting according to judgment that drawing said restrictions judges whether said permission has specified a set of a parameter and said permission has not specified a set of a parameter.

[Claim 9]It is judged whether it needs to mount an exemption mechanism in which said permission is [said processing of said permission] specific, A method according to claim 1 provided with adjusting said permission and said exemption restrictions in order to access a set of exemption restrictions and to draw said restrictions according to judgment that it needs to mount an exemption mechanism in which said permission is specific.
[Claim 10]Said adjustment with said permission and said exemption restrictions judges whether said exemption restrictions permit that said specific exemption mechanism is mounted to said mounting, A method according to claim 9 by which said exemption restrictions were provided with drawing said restrictions based on said exemption restrictions according to judgment that it permits that said specific exemption mechanism is mounted to said mounting.
[Claim 11]Said adjustment with said permission and said exemption restrictions judges whether said exemption restrictions permit that said specific exemption mechanism is mounted to said mounting, A method according to claim 9 provided with said exemption restrictions accessing a set of initial restriction according to judgment that it does not permit that said specific exemption mechanism is mounted to said mounting, and drawing said restrictions based on said initial restriction.
[Claim 12]said exemption restrictions -- two or more law -- a method according to claim 9 of being what is drawn by merging a policy and extracting restrictions restrictions from there.
[Claim 13]A method according to claim 1 provided with scrutinizing a Call Stack, in order to judge any application with which said judgment whether there is any permission given to said application required said mounting is.
[Claim 14]A method according to claim 13 further provided with said judgment whether there is any permission given to said application attesting said application.
[Claim 15]A device which determines restrictions in a system characterized by comprising the following as which mounting of specific service is required by application imposed on said mounting.
A mechanism in which it is judged whether there is any permission given to said application which requires said mounting.
A mechanism which processes said permission in order to lead a set of restrictions imposed on said mounting according to judgment that there is at least one permission given to said application.

[Claim 16]The device according to claim 15 which is what is processed so that said permission may serve as a restrictions degree with the smallest set of said restrictions.
[Claim 17]The device according to claim 15 further provided with a mechanism which accesses a set of initial restriction, and a mechanism to carry out in which said restrictions are drawn based on said initial restriction, according to judgment that there is no

permission given to said application.
[Claim 18]said initial restriction -- two or more law -- the device according to claim 17 which is what is drawn by merging a policy and extracting restrictions restrictions from there.
[Claim 19]The device comprising according to claim 15:
A mechanism in which said mechanism which processes said permission judges whether it is the permission in which said permission includes all.
A mechanism which prepares an index of being a thing without restriction of said mounting according to judgment that said permission is the permission which includes all.

[Claim 20]The device comprising according to claim 15:
A mechanism which said mechanism which processes said permission judges for whether said permission needs to mount an exemption mechanism.
A mechanism in which a set of said restrictions is led based on said permission according to judgment that said permission does not need to mount an exemption mechanism.

[Claim 21]The device comprising according to claim 20:
A mechanism in which said mechanism in which said restrictions are drawn judges whether said permission has specified a set of a parameter.
A mechanism in which a set of said restrictions is led based on said parameter according to judgment that said permission has specified a set of a parameter.

[Claim 22]The device comprising according to claim 20:
A mechanism in which said mechanism in which said restrictions are drawn judges whether said permission has specified a set of a parameter.
A mechanism which prepares an index of being a thing without restriction of said mounting according to judgment that said permission has not specified a set of a parameter.

[Claim 23]The device comprising according to claim 15:
A mechanism in which it is judged whether it needs to mount an exemption mechanism in which said permission is [said mechanism which processes said permission] specific.
A mechanism which accesses a set of exemption restrictions according to judgment that it needs to mount an exemption mechanism in which said permission is specific, and a mechanism in which said permission and said exemption restrictions are adjusted in order to draw said restrictions.

[Claim 24]The device comprising according to claim 23:
A mechanism in which said mechanism in which said permission and said exemption restrictions are adjusted judges whether said exemption restrictions permit that said

specific exemption mechanism is mounted to said mounting.
A mechanism in which said exemption restrictions draw said restrictions based on said exemption restrictions according to judgment that it permits that said specific exemption mechanism is mounted to said mounting.
[Claim 25]The device comprising according to claim 23:
A mechanism in which said mechanism in which said permission and said exemption restrictions are adjusted judges whether said exemption restrictions permit that said specific exemption mechanism is mounted to said mounting.
A mechanism in which said exemption restrictions access a set of initial restriction according to judgment that it does not permit that said specific exemption mechanism is mounted to said mounting, and a mechanism in which said restrictions are drawn based on said initial restriction.
[Claim 26]said exemption restrictions -- two or more law -- the device according to claim 23 which is what is drawn by merging a policy and extracting restrictions restrictions from there.
[Claim 27]The device according to claim 15 with which said mechanism in which it was judged whether there is any permission given to said application was provided with a mechanism in which a Call Stack is scrutinized in order to judge any application which required said mounting is.
[Claim 28]The device according to claim 27 with which said mechanism in which it was judged whether there is any permission given to said application was further provided with a mechanism which attests said application.
[Claim 29]When it is the medium characterized by comprising the following which stored a command and in which computer reading is possible and said command is executed by 1 or two or more processors, this -- what is operated so that restrictions which impose 1 or two or more processors on mounting of specific service of which it was required by application may be determined
A command whose 1 or two or more processors are operated so that it may judge whether there is any permission given to said application with which a medium in which said computer reading is possible requires said mounting.
A command which operates 1 or two or more processors so that said permission may be processed in order to lead a set of restrictions imposed on said mounting according to judgment that there is at least one permission given to said application.
[Claim 30]A medium which is what is processed so that said permission may serve as a restrictions degree with the smallest set of said restrictions and in which the computer reading according to claim 29 is possible.
[Claim 31]According to judgment that there is no permission given to said application, so that a set of initial restriction may be accessed, A medium which was further provided with

a command which operates 1 or two or more processors, and a command to which said restrictions are led based on said initial restriction, and which operates 1 or two or more processors so that it may carry out and in which the computer reading according to claim 29 is possible.

[Claim 32]said initial restriction -- two or more law -- a medium which is what is drawn by merging a policy and extracting restrictions restrictions from there and in which the computer reading according to claim 31 is possible.

[Claim 33]A medium in which the computer reading according to claim 29 is possible, comprising:

A command said command which operates 1 or two or more processors operates [command] 1 or two or more processors so that it may judge whether it is the permission in which said permission includes all so that said permission may be processed.

A command which operates 1 or two or more processors so that an index of being a thing without restriction of said mounting may be prepared according to judgment that said permission is the permission which includes all.

[Claim 34]A medium in which the computer reading according to claim 29 is possible, comprising:

A command whose 1 or two or more processors are operated so that said permission may be processed and said permission may judge whether said command which operates 1 or two or more processors needs to mount an exemption mechanism.

A command which operates 1 or two or more processors according to judgment that said permission does not need to mount an exemption mechanism so that a set of said restrictions may be led based on said permission.

[Claim 35]A medium in which the computer reading according to claim 34 is possible, comprising:

A command which operates 1 or two or more processors so that said restrictions may be drawn and said command which operates 1 or two or more processors may judge whether said permission has specified a set of a parameter.

A command which operates 1 or two or more processors according to judgment that said permission has specified a set of a parameter so that a set of said restrictions may be led based on said parameter.

[Claim 36]A medium in which the computer reading according to claim 34 is possible, comprising:

A command which operates 1 or two or more processors so that said restrictions may be drawn and said command which operates 1 or two or more processors may judge whether said permission has specified a set of a parameter.

A command which operates 1 or two or more processors so that an index of being a thing without restriction of said mounting may be prepared according to judgment that said permission has not specified a set of a parameter.

[Claim 37]A medium in which the computer reading according to claim 29 is possible, comprising:

A command which operates 1 or two or more processors so that it may judge whether it needs to mount an exemption mechanism in which said permission is [said command which operates 1 or two or more processors] specific so that said permission may be processed.

A command which operates 1 or two or more processors according to judgment that it needs to mount an exemption mechanism in which said permission is specific so that a set of exemption restrictions may be accessed, and a command which operates 1 or two or more processors so that said permission and said exemption restrictions may be adjusted in order to draw said restrictions.

[Claim 38]A medium in which the computer reading according to claim 37 is possible, comprising:

A command whose 1 or two or more processors said command which operates 1 or two or more processors operates so that said exemption restrictions may judge whether it permits that said specific exemption mechanism is mounted to said mounting so that said permission and said exemption restrictions may be adjusted.

A command which operates 1 or two or more processors so that said exemption restrictions may draw said restrictions based on said exemption restrictions according to judgment that it permits that said specific exemption mechanism is mounted to said mounting.

[Claim 39]A medium in which the computer reading according to claim 37 is possible, comprising:

A command whose 1 or two or more processors said command which operates 1 or two or more processors operates so that said exemption restrictions may judge whether it permits that said specific exemption mechanism is mounted to said mounting so that said permission and said exemption restrictions may be adjusted.

A command which operates 1 or two or more processors so that said exemption restrictions may access a set of initial restriction according to judgment that it does not permit that said specific exemption mechanism is mounted to said mounting, and a command which operates 1 or two or more processors so that said restrictions may be drawn based on said initial restriction.

[Claim 40]said exemption restrictions -- two or more law -- a medium which is what is drawn by merging a policy and extracting restrictions restrictions from there and in which the computer reading according to claim 37 is possible.

[Claim 41]So that it may judge whether there is any permission given to said application, A

medium by which said command which operates 1 or two or more processors was provided with a command which operates 1 or two or more processors so that a Call Stack might be scrutinized, in order to judge any application which required said mounting is and in which the computer reading according to claim 29 is possible.

[Claim 42]So that it may judge whether there is any permission given to said application, A medium by which said command which operates 1 or two or more processors was further provided with a command which operates 1 or two or more processors so that said application might be attested and in which the computer reading according to claim 41 is possible.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention]Especially this invention relates to the mechanism for determining the restrictions imposed on mounting of the service demanded by application about a computer system.

[0002]

[Description of the Prior Art]For years, the U.S. Department of Commerce regulated export of the computer program or application containing a data encryption algorithm, and has forbidden depending on the case. The computer program which is using the encryption algorithm using the encryption key more than the fixed number of bits as a present principle cannot be exported (the length of the key which can be specified is peculiar to an algorithm). There is also an exception in this rule. One of the exceptions can increase the length of a key, i.e., the encryption strength of a program, by the case where the exemption mechanism is adopted, depending on the case. There are key escrow (key escrow), the key recovery (key recovery), and the key weakening (key weakening) in the example of an exemption mechanism. The length of a key can be enlarged depending on the kind of program. For example, the application for a medical institution and financial institutions enlarges the length of a key, and the present regulation permits what the safety of application is improved for (it is used for protection of advanced security data). While there is blessed application with larger tolerance level than other applications, export control is needed for all encryption applications.

[0003]

[Problem(s) to be Solved by the Invention]These regulations are applied also to the program which it is not only applied to the program which is using the encryption algorithm directly, but has an interface to the program which is using the encryption algorithm directly. The "framework" program which provides the infrastructure for performing the interaction

during various programs smoothly is included in a program. Although the framework itself does not mount any encryption algorithms, it is permissible that one or more programs which mount the encryption algorithm interface to a framework, or carry out "plug-in" to a framework. It is Java Cryptography Extension of Java Platform by [of the example of such a framework] Sun Microsystems, Inc. of California and Palo Alto in one. In permitting that a framework carries out "plug-in" of the cipher device style to a framework, export control is needed for the framework itself. This means that a framework needs to guarantee that all the export control is protected irrespective of code mounting [BURAGUIN / mounting / the framework], in order to be made to be possible [export]. In order to offer this guarantee, a framework needs to restrain code mounting with one of mechanisms.

[0004]

[Means for Solving the Problem]If this invention is followed, a mechanism in which mounting which imposes restrictions on service and by which custom-made ** was carried out is built dynamically is provided. For the purpose of this invention, service is defined as a broad sense and includes all functions provided to a demand or this application by application including encryption/decoding function (however, not limited to this). In one embodiment of this invention, an invention is realized within a system provided with general mounting and a framework of application and specific service.

[0005]A framework receives a demand of mounting of specific service, for example, mounting of a specific encryption algorithm, from application. When restrictions of a framework exist corresponding to this, restrictions which need to be imposed on demanded mounting are determined. In one embodiment, when these restrictions are determined and there is this permission by judging whether a framework has the permission given to said application, in order to lead a set of restrictions imposed on said mounting, this permission is processed. In one embodiment, this permission is processed so that a set of restrictions led may serve as the smallest restrictions degree. If these restrictions are determined, a framework will build demanded mounting dynamically. In one embodiment, demanded mounting is built so that it may incorporate enforcement logic which imposes restrictions on general mounting of said service, said restrictions, and said general mounting. Since demanded mounting was built for said application, it was customized for [the] applications. Therefore, this mounting is called mounting by which custom-made ** was carried out.

[0006]After mounting by which custom-made ** was carried out is built dynamically, a framework provides application with mounting by which custom-made ** was carried out. Then, application calls directly mounting by which custom-made ** was carried out for service. Since restrictions and enforcement logic for imposing it are included in mounting by which custom-made ** was carried out, the application does not need to act on a framework and mutual further. The mounting itself by which custom-made ** was carried out will provide service, and restrictions will be added certainly. Thus, according to

dynamic construction of mounting by which custom-made ** was carried out, restrictions which needs a framework for service provided to application can be added.

[0007]

[Embodiment of the Invention]The block diagram of the system 100 by which one of the embodiments of this invention is realized is shown in drawing 1. The framework 102 for performing smoothly the interaction between 1, two or more applications 104 and 1 or two or more general mounting 106, the set 108 of the specified restriction, and various kinds of components is included in this system 100. The application 104 requires mounting of service of the framework 102, and receives it. Here, various kinds of applications or programs may be sufficient as the application 104, and it contains a Java applet, Java application, the application (not limited to these) compiled native, etc. For the purpose of this invention, the term of "service" is defined as a broad sense, and includes all the functions provided to a demand or this application by application including encryption/decoding function (however, not limited to this).

[0008]The application 104 specifies the kind of service for mounting which it desires, when requiring mounting of the framework 102. For example, the application 104 can require mounting of the "Blowfish" encryption algorithm. Corresponding to this, the framework 102 provides the applications 104 which have advanced the demand with mounting of the demand service by which custom-made ** was carried out at the application 104. Restrictions of the service which a framework provides are included in mounting which is provided by the framework 102 and by which custom-made ** was carried out. These restrictions are determined based on the set of the specified restriction 108, and the permission 110 observed in the application 104 which has advanced the demand if it was so that it may mention later.

[0009]The general mounting 106 expresses mounting of the service which "plug-in" is carried out to the framework 102, or interfaces. Each of the general mounting 106 realizes service of a specific kind. For example, one general mounting mounts the "Blowfish" encryption algorithm, and another mounting mounts a DES encryption algorithm simultaneously. Each of the general mounting 106 is not restrained. That is, even if the restriction 108 or the permission 110 exists, mounting 106 general the very thing is not barred by restrictions. By this, when the general mounting 106 is mounting of an encryption algorithm, an encryption algorithm can be set as perfect intensity. The framework 102 guarantees that suitable restrictions are added to the service provided for the application 104, and it is not the general mounting 106 so that it may explain below.

[0010]In the system 100, the framework 102 is a component which adjusts the whole operation of the system 100. The flow chart showing general operation of the framework 102 is shown in drawing 2. The framework 102 operates by receiving the demand of mounting (for example, mounting of a Blowfish encryption algorithm) of service of a specific

kind from the application 104 as shown in drawing 2 (202). When restrictions exist corresponding to this, the framework 102 judges restrictions required for demanded mounting (204). In one embodiment of this invention, if there is the framework 102 about the specified restriction 108, it will judge restrictions by adjusting with the permission 110 observed in the application 104 which has advanced the demand. And in one embodiment of this invention, the framework 102 tends to add minimum restrictions as much as possible. In other words, in consideration of the permission 110 and the restriction 108, the framework 102 is tried so that generously as much as possible.

[0011]If restrictions are decided, the framework 102 will build demanded mounting dynamically (206). In one embodiment of this invention, demanded mounting is constituted by finding the general related mounting 106 which mounts the demanded kind (for example, general mounting 106 which realizes a Blowfish encryption algorithm) of service. If this is found, this general related mounting 106 will be included in demanded mounting with the restrictions determined before. The set of enforcement logic is also built into this demanded mounting. This enforcement logic guarantees that these restrictions are imposed on the general related mounting 106. Therefore, in spite of not attaching restrictions to mounting 106 general related the very thing, enforcement logic serves as suitable restrictions applied to the general related mounting 106. General related mounting, the restrictions incorporated here, and enforcement logic are used, and the demanded construction of mounting is completed. Since demanded mounting incorporates construction, i.e., the demanded restrictions peculiar to application, specially to the demanded applications 104, demanded mounting can be seen to the demanded applications 104 as mounting by which custom-made ** was carried out.

[0012]Construction of mounting by which custom-made ** was carried out will pass this mounting to the demanded application 104 (208). Then, the application 104 requires service of mounting by which custom-made ** was carried out directly. Since the enforcement logic for imposing restrictions and restrictions on mounting by which custom-made ** was carried out is incorporated, it becomes unnecessary for the application 104 to act on the framework 102 and mutual further. The mounting itself by which custom-made ** was carried out provides service, and restrictions come to be certainly added to service. By building dynamically mounting by which custom-made ** was carried out like this method, the restrictions which needs the framework 102 for the service provided for the application 104 are added certainly.

[0013]The above-mentioned explanation provides the general outline of this invention. Drawing 3 explains one embodiment of this invention in detail. By the following explanation, this invention is required and the service provided is explained in relation to object-oriented mounting which is cipher service. Note that this is used only for the purpose of explanation. This invention is not limited to the range of explanation. If it says

appropriately, this invention will generally be applied to all kinds of programming environment, and all the kinds to which restrictions need to be added of service.

[0014]The details of the framework 102 are shown in drawing 3 and drawing 4. The framework 102 is provided with the application programming interface (API) 302, the service provider interface (SPI) 304, and the core 320 as shown in the figure. API302 expresses the resource which the application 104 can call directly. API302 is provided with Cipher object classes 306 and ExemptionMechanism object classes 308 in one embodiment of this invention. It is mixed with other methods and Cipher object classes 306 is provided with a GetInstance method and an Init method. A GetInstance method is a method called by the application 104, when application requires mounting of service. Corresponding to this method call, the instance of Cipher object classes 306 is built and it is returned to the application 104 currently called. Custom-made ** of the returned Cipher instance is carried out for [which is called] applications, and it contains the enforcement logic for adding these restrictions to restrictions and the service which can provide a Cipher instance. If a Cipher instance is returned, call appearance of the method of a Cipher instance will be directly carried out by the application 104. One of the methods which need to be called by the application 104 currently called is an Init method. This method initializes a Cipher instance and enables it to operate a Cipher instance. The Init method operates as enforcement logic for adding restrictions to a Cipher instance. A GetInstance method and an Init method are mentioned later in detail.

[0015]As already explained, when 1 or two or more exemption mechanisms (a key escrow, a key recovery, or key weakening) are mounted, the encryption algorithm (for example, key length was lengthened) which strengthened encryption strength depending on the case can be realized. When the exemption mechanism is mounted, ExemptionMechanism object classes 308 operates. This class provides two or more methods which can be called. In order that these methods may call the function of a specific exemption mechanism (for example, a key recovery block is generated when an exemption mechanism is a key recovery), It is called in order to judge whether required operation was performed (for example, was the key recovery block generated or not?). Object classes 306 and 308 of API302 is explained in detail later.

[0016]SPI304 provides an interface required for a service provider, and carries out plug-in of the service mounting of a service provider to the framework 102. SPI304 is provided with SPI304 object classes corresponding to each API302 object classes in one embodiment of this invention. That is, CipherSpi object classes 310 of SPI304 corresponding to Cipher object classes 306 of API302 exists. And ExemptionMechanismSpi object classes 312 of SPI304 corresponding to ExemptionMechanism object classes 308 of API302 exists. This correspondence of 1 to 1 makes it easy to map the method of the API classes 306 and 308 in the method of the SPI classes 310 and 312. This importance is mentioned later in detail.

SPI object classes 310 and 312 is abstract object classes, and while the method which must be mounted by the class is shown, it means that object classes itself provides no mounting of these methods. A service provider takes charge of offer of mounting. In order to provide the mounting 106 of service, a service provider subclasses one of the object classes of SPI304, and mounting is prepared for [all the] the method in which the SPI class was defined as the subclass. Thus, the general mounting 106 shown in drawing 3 becomes a subclass of object classes 310 and 312 of SPI304. Each of the general mounting 106 can also mount service of a different kind. Each of (for example, a Blowfish encryption algorithm is mounted, and a DES encryption algorithm is mounted simultaneously, and a key recovery exemption mechanism can be mounted simultaneously) and the general mounting 106 can be realized without receiving no restrictions. General mounting 106 can also be carried out to come at mounting (for example, unrestricted in the length of an encryption key) of the maximum intensity. The core 320 of the framework 102 is provided with JCESecurity object classes 314 and JCESecurityManager object classes 316. In one embodiment of this invention, these object classes 314 and 316 is package private life, and cannot carry out direct access of the application 104. A JCESecurity class is provided with a GetImpl method and a JCESecurityManager class is provided with a GetCryptoPermission method as shown in drawing 3. These methods are called as a result of the call of the GetInstance method of the Cipher class 306, collaborate and do the work of a large number required for dynamic construction of mounting by which custom-made ** was carried out. If the contents of all the systems are understood, you can understand well the function performed by these methods. Therefore, next, in order to understand all the inventions smoothly, the flow chart of drawing 5 and drawing 6 explains all the operations of a system.

[0017]When it needs mounting of specific cipher service, the application 104 is calling the GetInstance method of Cipher object classes 306, and advances the demand of mounting. In this call, application specifies the kind of service which is demanding mounting. In one embodiment of this invention, the kind of service becomes an encryption algorithm name like Blowfish, for example. The Cipher class 306 receives this demand (404), and calls the function of a GetInstance method. Corresponding to this, a GetInstance method calls the GetImpl method of the JCESecurity class 314. A GetImpl method performs two or more important functions. This method judges whether it is usable in the general mounting 106 which mounts service of the demanded kind first (408). For example, it is judged whether either of the general mounting 106 mounts the Blowfish encryption algorithm. When the suitable general mounting 106 is not found, an error message is returned to the application 104 which returns an error message to a GetInstance method (412), next is called. On the contrary, when the general mounting 106 which mounts demanded service is found, a GetImpl method keeps on whether attesting found general mounting, and judges (416). although the method of performing this attestation is mentioned later in detail, attestation

is performed here using a digital signature verification mechanism -- only -- it explains.
[0018]When a GetImpl method judges that the general mounting is not attested, it is judged whether the general mounting 106 which can offer service demanded exists else (420). When the mounting 106 general to others does not exist, a GetImpl method returns an error message to the application 104 which returns an error message to a GetInstance method (424), next is called. When the general mounting 106 which can offer service demanded exists in others, a GetImpl method judges whether it returns to the process 416 and general new mounting is attested. It continues until this processing is judged that the attested general mounting 106 which can offer service which attested mounting was found or was demanded does not exist.
[0019]When the general mounting 106 (this mounting will be called related mounting) with which the demanded service was attested is found, A GetImpl method instantiates related mounting and generates the instance (namely, CipherSPI instance) of mounting (428). Then, a GetImpl method judges whether it is necessary to add a certain restriction to the instance of mounting (432). In one embodiment of this invention, this judgment is made by judging whether it is set up for the operation with the internal framework 102, or global operation. Since export control is not applied when being set up so that a framework may be restricted to the use in the country, it is not necessary to add restrictions. The possibility of restriction is taken into consideration when set up, perform operation with the global framework 102 on the other hand.
[0020]In order to judge the restrictions added to the instance of mounting, (436) and a GetImpl method call the GetCryptoPermission method of the JCESecurityManager class 316. The important function of a GetCryptoPermission method is adjusting the specified restriction 108 and the permission 110 observed in the application 104 currently called if it was, and leading the set of restrictions. The set of these restrictions is returned to a GetImpl method by the GetCryptoPermission method. And in one embodiment of this invention in the set of these restrictions. Some cryptographic parameters, such as a demanded name of an encryption algorithm, a name of the exemption mechanism which needs to be imposed (supposing it exists), the maximum key length, the maximum execution repetition number (required for algorithms, such as RC5) of a code which are used, are contained. A GetImpl method will judge whether the exemption mechanism is specified within these restrictions, if these restrictions are received (440). When the exemption mechanism is not specified within restrictions, he follows a GetImpl method to the process 448.
[0021]However, when the exemption mechanism is specified, a GetImpl method continues and generates the instance of the specified exemption mechanism. In one embodiment of this invention, this calls the GetInstance method of the ExemptionMechanism class 308, and is attained by telling the name of an exemption mechanism. To this call, the

GetInstance method of the ExemptionMechanism class 308, The GetImpl method of the JCESecurity class 314 is called (this call wants to be cautious of becoming the call of the 2nd of a GetImpl method). Corresponding to this, a GetImpl method searches the effective general mounting 106 which mounts the specified exemption mechanism, instantiates the general mounting 106 (444), and generates an ExemptionMechanismSpi instance. Then, a GetImpl method returns an ExemptionMechanismSpi instance to the GetInstance method of the ExemptionMechanism class 308 (this is the return from the call of the 2nd of a GetImpl method).

[0022]Next, the GetInstance method of the ExemptionMechanism class 308, The constructor of the ExemptionMechanism class 308 is called and the ExemptionMechanismSpi instance returned from the GetImpl method is passed to a constructor. When called, a constructor instantiates the ExemptionMechanism class 308 and generates an ExemptionMechanism instance. Next, a constructor encapsulates an ExemptionMechanismSpi instance in an ExemptionMechanism instance. It maps in a method [method / of an ExemptionMechanism instance / constructor / in that case / instance / ExemptionMechanismSpi]. The Init method of an ExemptionMechanism instance is mapped by the EngineInit method of an ExemptionMechanismSpi instance in one embodiment of this invention, A GenExemptionBlob method is mapped by the EngineGenExemptionBlob method. As for this mapping, a call in the method of an ExemptionMechanism instance is sent to the right method of an ExemptionMechanismSpi instance. If an ExemptionMechanismSpi instance is encapsulated in an ExemptionMechanism instance, instantiation of an ExemptionMechanism instance will be completed.

[0023]Then, a GetImpl method returns to the GetInstance method of the Cipher class 306 (this). A certain GetInstance method is provided with the instance of mounting, the set of restrictions, and (supposing it is) an ExemptionMechanism instance by the return from the 1st call of a GetImpl method. Next, the GetInstance method of the Cipher class 306, The constructor of the Cipher class 306 is called and the instance of mounting received from the GetImpl method to the constructor, the set of restrictions, and (supposing it is) an ExemptionMechanism instance are passed. Corresponding to this, a constructor instantiates the Cipher class 306 (448) and generates a Cipher instance. Next, a constructor encapsulates the instance of mounting, the set of restrictions, and (supposing it is) an ExemptionMechanism instance in a Cipher instance (452). That is, the Cipher instance operates as a "wrapper" object. A constructor is mapped in the instance method of mounting corresponding to the case where the instance of mounting is encapsulated to a Cipher instance, for the method of a Cipher instance. In one embodiment of this invention, the Init method of a Cipher instance, It is mapped by the EngineInit method of the instance of mounting, a Update method is mapped by the EngineUpdate method, and a DoFinal

method is mapped by the EngineDoFinal method. This mapping is sent to the method of the instance of right mounting of a call in the method of a Cipher instance. Since mounting of these methods is provided by the instance of mounting, it becomes such. If encapsulation processing is completed, a constructor will return to the GetInstance method of the Cipher class 306. Next, a GetInstance method returns to the application 104 currently called, and provides the Cipher instance newly built by the application 104 (456). Then, the application 104 currently called can call the method of a Cipher instance directly.

[0024]In one embodiment of this invention, one of the first methods to which the application 104 currently called needs to call a Cipher instance is an Init method. This method initializes a Cipher instance and prepares an Init method for the usual operation. While calling this method, the application 104 currently called provides the set of initialization parameters. In one embodiment of this invention, the encryption key used for encryption and other arbitrary cryptographic parameters which have specified the attribute peculiar to algorithms, such as a repetition number of a code, are contained in these parameters (when a specific encryption algorithm needs).

[0025]When an Init method is called, an Init method compares the initialization parameters passed by the application 104 currently called with the restrictions encapsulated in the Cipher instance. When initialization parameters are a level of restrictions, or less than it, an Init method is passed to the EngineInit method of the instance of mounting of initialization parameters, and enables it to initialize the instance of mounting. After the instance of mounting is initialized, operation of a Cipher instance is attained. thus, the application 104 which is calling the Update method and DoFinal method of the Cipher instance in order to perform operation of encryption/decryption -- therefore, it can call. However, when it is judged that the level of restrictions with which the initialization parameters passed by the application 104 which the Init method is calling were encapsulated was exceeded. An Init method is made not to be passed to the EngineInit method of the instance of mounting of initialization parameters. It is made not to initialize by it, the instance, i.e., the Cipher instance, of mounting. When a Cipher instance is not initialized, it becomes impossible for the Cipher instance to operate normally. Thus, an Init method is prevented from operating a Cipher instance effectively by not initializing. By this method, the encapsulated restrictions commit an Init method as enforcement logic which ensures being imposed on the instance of mounting.

[0026]When an ExemptionMechanism instance is encapsulated in a Cipher instance, the Init method of the Cipher class 306 performs an additional function. The function ensures performing required operation, before an ExemptionMechanism instance is called correctly and performs a data encryption with the application 104. For example, when an exemption mechanism is a key recovery, before enciphering data, it is necessary to call an ExemptionMechanism instance, and to generate and save a key recovery block. In order to

ensure what required operation was performed for by the ExemptionMechanism instance, an Init method calls the IsCryptoAllowed method of an ExemptionMechanism instance. In one embodiment of this invention, an ExemptionMechanism instance, Information is held in it about whether the GenExemptionBlob method was called (the ExemptionMechanism instance is the origin by which operation of a required exemption mechanism is performed). An IsCryptoAllowed method is called and this information can access it. Operation which needs this IsCryptoAllowed method was performed (.). That is, when what the GenExemptionBlob method was called for is shown, an Init method enables it to initialize, the instance, i.e., the Cipher instance, of mounting. Since it keeps an Init method from the ability of initialization to do when required operation is not performed, the Cipher instance can operate no longer. Therefore, an Init method not only adds restrictions to the instance of mounting, but ensures that an exemption mechanism is applied.

[0027]As mentioned above, it is a GetCryptoPermission method of the JCESecurityManager class 316 which judges the restrictions added to the service provided by the Cipher instance. These restrictions will be determined as the specified restriction 108 based on the permission 110 observed in the application 104 currently called, if it is. Although one embodiment of a GetCryptoPermission method is described below, before describing an embodiment in detail, in order to understand all this inventions smoothly, the restriction 108 and the permission 110 are explained briefly.

[0028]In one embodiment of this invention, the restriction 108 comprises two-set restriction of default configuration and exemption setting out. Fundamentally, default configuration specifies the initial restriction which needs to be added to an encryption algorithm, when the exemption mechanism is not mounted. And when the exemption mechanism is mounted, the restriction which needs to be added to an encryption algorithm is specified as exemption setting out. Generally, when the exemption mechanism is mounted, a firm cryptographic parameter can be used. In one embodiment of this invention, restriction of both setting out is due to the law and regulation which are applied.

[0029]Each setting out (default configuration or exemption setting out) of restriction comprises 0 or one or more entries. Some restrictions added to a specific encryption algorithm and its algorithm are specified as each entry. The entry of each setting out about restriction is having the same structure. In one embodiment of this invention, each entry comprises the field or the information container which saves the following information.

(1) Restriction of a code peculiar to the algorithm of others, such as the maximum repetition number of a code, by which an encryption algorithm name, an identifier (2) exemption mechanism name or the identifier (3) maximum key length, and (4) execution are carried out [0030]Because of the purpose of this invention, any desirable forms are possible for an entry. For example, it can mount as an object with the required information encapsulated in it in each entry, and each entry can also be made into the combination of a text in a file. As

long as right information is provided, any desirable thing forms can be used.
[0031]The example of the default configuration of restriction and exemption setting out is shown in drawing 7. In the entry of default configuration, it is cautious of an exemption mechanism specifying neither but certainly specifying an exemption mechanism by the entry of exemption setting out. Since default configuration specifies the restrictions added when the exemption mechanism is not mounted and exemption setting out specifies the restrictions added when the exemption mechanism is mounted, it becomes such.
[0032]The interpretation of the default configuration of restriction is easy. Fundamentally, each entry expresses the maximum cryptographic parameter about a specific encryption algorithm. Therefore, with a Blowfish algorithm, the 128-bit maximum key length is used like drawing 7. Similarly, in RC5 algorithm, the 64-bit maximum key length and the maximum repetition number of 10 times of codes are used. The interpretation of exemption setting out is easy almost similarly. Fundamentally, the 1st entry of exemption setting out shows that the maximum key length is made to increase and it is made to 256 bits, when the key recovery exemption mechanism is mounted with the Blowfish algorithm. Similarly, it is shown that the 2nd entry makes the maximum key length increase, and is made to 256 bits when the key escrow exemption mechanism is mounted with the Blowfish algorithm. Note that the same algorithm name (in this case, Blowfish) can be described for two or more entries in exemption setting out. If the exemption mechanisms specified as these entries differ, the same algorithm name can be described.
[0033]The restriction 108 specified is a part of only factor taken into consideration by the determination of the restrictions added to a Cipher instance. The permission 110 observed in the application 104 currently called exists, then another factor is them. As explained above, firm cryptography can be used for a kind like the application for a medical institution and financial institutions of application compared with other applications. The authority to use firm cryptography with the application for a medical institution and financial institutions or other applications is reflected in the permission 110 observed in application. In one embodiment of this invention, the permission 110 takes one of two or more forms. The 1st form is CryptoAllPermission information. When application is given CryptoAllPermission, all the permissions in which the meaning is possible to application will be given. In other words, application is not restricted. This enables the highest permission that can be accepted, therefore is observed in the application of a **** small number.
[0034]Permission lower than this which is observed in application is permission for strengthening the encryption strength of a specific cryptographic algorithm, or mounting indefinitely. In one embodiment of this invention, this kind of permission specifies the combination (for example, the maximum key length) of a specific algorithm name (for example, Blowfish) and arbitrary maximum parameters. When the combination of a

maximum parameter is specified, an encryption algorithm may be mounted on the level of the maximum parameter specified. When the combination of a maximum parameter is not specified, an encryption algorithm may be mounted on arbitrary levels (that is, an algorithm is not restrained). Thus, when Blowfish is specified with the 128-bit maximum key length by permission, the application can use a Blowfish encryption algorithm by the 128-bit maximum key length. Only in the case that Blowfish is specified by permission, the application can use a Blowfish encryption algorithm, without being restricted to the length of a key. Until now, only the maximum key length was explained about the maximum parameter. The maximum parameter should be cautious of the ability of another parameters, such as the maximum repetition number of a code, to be included. Such an another parameter can also require encryption algorithms, such as RC5, therefore can also include them into a maximum parameter.

[0035]Other permissions observed in application are permissions for mounting a specific exemption mechanism in a specific encryption algorithm (for example, key recovery which uses Blowfish). If an exemption mechanism is mounted as mentioned above, application can usually use a firm cryptographic parameter (for example, long key length). Thus, the permission which mounts an exemption mechanism raises encryption strength dramatically. It is explained below by that permission is actually dependent on the contents of the restriction 108 whether it lends, and there is nothing or it is usable in mounting of an exemption mechanism. At this point, application must be noticed also about it being possible for two or more permissions to be accepted. For example, application can also mount two or more kinds of exemption mechanisms. One application can have two or more permissions accepted that case and in the case of others.

[0036]Next, the flow chart of drawing 8 explains operation of the GetCryptoPermission method of the JCESecurityManager class 316 based on such background information. A GetCryptoPermission method receives the set of the parameter containing the encryption algorithm name (for example, Blowfish) demanded by the application 104 currently called, when it is called. Corresponding to a call, a GetCryptoPermission method determines the application 104 currently called first (604). That is, a GetCryptoPermission method determines the application 104 which called the GetInstance method which became a cause by which a GetCryptoPermission method was called. In one embodiment of this invention, a GetCryptoPermission method makes this decision by examining a Call Stack in detail. This traces a call order and returns from a GetCryptoPermission method to a GetImpl method, Next, it carries out by returning to a GetInstance method and returning to the application 104 which is calling the GetInstance method first next. Like this method, by examining a Call Stack in detail, the beginning calls and a GetCryptoPermission method can determine the application 104.

[0037]Determination of the application 104 currently called will determine whether the

application 104 currently called has a certain effective permission observed in it (608). In one embodiment of this invention, this is performed by determining first whether a certain permission was primarily given to the application 104. In one embodiment of this invention, this decision is made by checking the file related to the application 104 and checking whether a certain permission is included in this. By a Java programming environment, the file of application is included in a JAR file and uses this JAR file for the check of permission in this environment.

[0038]When a certain permission is found, verification processing is performed and it is guaranteed that permission is effective. In one embodiment of this invention, this verification is performed using a digital signature. Specifically, the JAR file by which the digital signature was carried out exists in the arbitrary applications 104 including 1 or two or more permissions. The source of the application 104 is trusted and this digital signature ensures that the contents of the application 104 were not changed. When this digital signature is verified, it means that the permission included in a JAR file is effective. Permission is invalid when this digital signature is not verified. A GetCryptoPermission method performs this verification using a digital signature verification mechanism. A digital signature verification mechanism suitable because of the purpose of this invention and effective can also be used.

[0039]When it is judged that the application 104 which the GetCryptoPermission method is calling does not have effective permission, A GetCryptoPermission method determines the restrictions added to a Cipher instance based on restriction of the default configuration of restriction (612). Specifically, a GetCryptoPermission method searches the default configuration entry of the entry of the same algorithm name as the encryption algorithm demanded by the application 104 currently called. After the entry is found, restrictions are drawn from the restriction (for example, the maximum key length and other restrictions) specified in the entry. For example, as for restrictions, the maximum key length will be set to 128-bit Blowfish like the example of drawing 7, when the application 104 currently called is demanding mounting of a Blowfish algorithm. After restrictions are decided, restrictions are returned to the GetImpl method of the JCESecurity class 314 by the GetCryptoPermission method (616).

[0040]When it is judged that the application 104 which returns to the process 608 and the GetCryptoPermission method is calling has 1 or two or more effective permissions, A GetCryptoPermission method determines whether either of these permissions is CryptoAllPermission (620). In CryptoAllPermission, the application 104 is not restricted. In that case, a GetCryptoPermission method returns directions without restrictions to a GetImpl method (624). However, when all permissions are not CryptoAllPermission(s), he follows a GetCryptoPermission method to the process 628.

[0041]When it processes to the process 628, the application 104 has 1 or two or more

effective permissions, and it turns out that no these permissions are CryptoAllPermission(s). Therefore, it means that permission becomes either of two kinds of the following.

(1) The kind (namely, kind which specifies the set of a specific encryption algorithm and an optional maximum parameter) which does not require the exemption mechanism which should be added, or the kind which requires the exemption mechanism which should be (2) Added (namely, kind which specifies an exemption mechanism with a specific encryption algorithm)

[0042]At the process 628, a GetCryptoPermission method determines whether either of the permissions is a thing of the kind which does not require the exemption mechanism which should be added. When either of the permissions is this kind, it determines about whether that permission is applicable to each of those permissions (632). Permission is applicable when the encryption algorithm specified as permission is the same as the encryption algorithm demanded by the application 104. For example, one permission is applied when the application 104 is demanding mounting of a Blowfish algorithm, and the encryption algorithm specified as permission specifies the Blowfish algorithm. A maximum of one permission is applied in one embodiment of this invention. When determining that one of the permissions of a GetCryptoPermission method is applied, a GetCryptoPermission method, The restrictions added to a Cipher instance based on the maximum parameter (supposing it exists) specified as permission are determined. That is, when the set of a maximum parameter is specified as permission, restrictions are determined based on the specified maximum parameter. When the combination of the maximum parameter is not specified, restrictions become unrestricted and an encryption algorithm is not restrained. After restrictions are determined, restrictions are returned to the GetImpl method of the JCESecurity class 314 by the GetCryptoPermission method (636).

[0043]any of the permission which does not require the exemption mechanism in which it returns to the process 632 and a GetCryptoPermission method should be added -- although -- when being inapplicable is determined, it progresses to the process 640. At the process 640, a GetCryptoPermission method determines whether either of the permissions given to the application 104 is a kind which requires the exemption mechanism which should be added. When such permission is not found, a GetCryptoPermission method determines the restrictions added to a Cipher instance using the default configuration of restrictions (644). The method of determining restrictions is the same as the method explained in relation to the above-mentioned process 612. After restrictions are decided, restrictions are returned to a GetImpl method by the GetCryptoPermission method (648).

[0044]On the other hand, when a GetCryptoPermission method determines that at least one of the permissions observed in the application 104 is a kind which requires the exemption mechanism which should be added, it progresses to the process 652. At the

process 652, a GetCryptoPermission method determines whether either of the permissions which require the exemption mechanism which should be added is applicable. Specifically a GetCryptoPermission method, Mounting of the exemption mechanism which could use the set of specific its encryption algorithm / exemption mechanism, or was specified determines whether to be usable or not to each of the permission which can be applied to the encryption algorithm as which which of these permissions is demanded, or is applied. When you perform these functions, refer to the exemption setting out of restriction for a GetCryptoPermission method. He can understand these operations well by illustration.
[0045]The encryption algorithm demanded is a Blowfish algorithm and application presupposes that the following two kinds of permissions are accepted.
(1) Blowfish, key weakening, and (2) Blowfish and a key recovery [0046]Exemption setting out of the restriction furthermore shown in drawing 7 is assumed. In this example both, since permission relates to Blowfish, both permissions apply to the demanded application. Therefore, both both permissions will be processed and it is begun to process them from the 1st permission. Key weakening can be made to use the 1st permission with Blowfish. In order that this permission may determine whether to be usable or not, a GetCryptoPermission method searches exemption setting out with this combination of an entry. Although two entries of Blowfish exist in exemption setting out, neither of these entries specifies key weakening as an exemption mechanism. Therefore, this permission cannot be used or applied by that the combination of Blowfish which can use exemption setting out of restriction, and key weakening is not clearly shown to be and which is twisted.
[0047]In this case, he follows a GetCryptoPermission method to processing of permission of the next which permits the key recovery used with Blowfish. This permission searches the entry of the same method as the beginning, i.e., exemption setting out, and is processed. This time, the entry which can use the combination as which Blowfish and a key recovery were specified is found. As a result, use or application of this permission is attained. However, an inquiry does not finish there. A GetCryptoPermission method determines whether to be usable in effective mounting of the specified exemption mechanism (this example key recovery), before accepting use of this permission. And this permission is not applied when not usable in mounting. When making this decision, a GetCryptoPermission method searches the effective general mounting 106 (drawing 4) which mounts the specified exemption mechanism. It will understand whether the GetCryptoPermission method can apply either of the accepted permissions by the end (652) of this processing.
[0048]When a GetCryptoPermission method determines that permission is applicable, a GetCryptoPermission method uses not the default configuration of restriction but exemption setting out, and determines the restrictions added to a Cipher instance (656). Specifically, a GetCryptoPermission method draws restrictions from the entry of exemption setting out with the same algorithm name as this permission, and an exemption

mechanism. This entry is an entry of the beginning of exemption setting out in the target example, and those restrictions are Blowfish which the maximum key length equipped with a 256-bit key recovery. After these restrictions are decided, restrictions are returned to the GetImpl method of the JCESecurity class 314 by the GetCryptoPermission method (660). The entry of exemption setting out enables it to usually use a cryptographic parameter firmer than default configuration, as mentioned above. Therefore, a GetCryptoPermission method raises the encryption strength of a Cipher instance by drawing restrictions of exemption setting out.

[0049]When it returns to the process 652 and neither of the permissions can be applied, a GetCryptoPermission method uses the default configuration of restriction and determines the restrictions added to a Cipher instance (644). The method of determining restrictions is the same as the method explained above in relation to the process 612. Therefore, the application 104 is dealt with like the case where application is not given permission at all. After restrictions are decided, restrictions are returned to a GetImpl method by the GetCryptoPermission method (648). Like the explained method, a GetCryptoPermission method determines the restrictions added to a Cipher instance. A GetCryptoPermission method tends to give the encryption strength maximum with all restrictions given to a Cipher instance by using initial restriction, when it is going to apply permission first, next applies neither of the permissions. In other words, a GetCryptoPermission method tends to add restrictions of a minimum level.

[0050]As mentioned above, the default configuration of the restriction including all the sets (drawing 1) of the restriction 108 and exemption setting out are due to applicable law and regulation. In one embodiment of this invention, they are drawn based on the following two laws and regulations at least.

(1) The U.S. exporting method and (2) local method (law of the country where the framework 102 is imported, or the area)

[0051]Since the sets of these laws differ in almost all cases, regulated treatment is performed in order [which is in agreement with the set of both laws] to lead the set of one restriction. In one embodiment of this invention, this adjustment is performed using merge processing. The set of two laws is merged, and generates a set as a result of the restriction 108, and, specifically, merge is performed by the way the obtained restriction 108 includes the restriction when the set of two laws was restrained most. By choosing the restriction restrained most, merge processing guarantees that the obtained restriction 108 follows the set of both laws.

[0052]Drawing 9 expresses the outline of merge processing. The U.S. exporting method 702 comprises the default component 706 and the exemption component 708 as shown in the figure. Similarly, the local method 704 comprises the default component 710 and the exemption component 712. The default components 706 and 710 specify the default

restriction applied to an encryption algorithm, when the exemption mechanism is not mounted. And the exemption components 708 and 712 specify restriction in case the exemption mechanism is mounted. In one embodiment of this invention, the default components 706 and 710 and the exemption components 708 and 712 are holding the same form as the default configuration 714 of restriction and the exemption setting out 716 which were explained in relation to drawing 7 above. That is, each components 706, 710, 708, and 712 comprise 0 or an entry beyond it. Each entry is provided with the field or the container for saving the following.

(1) An encryption algorithm name, an identifier (2) exemption mechanism name or the identifier (3) maximum key length, and code restrictions of (4) others [0053]In order to draw the obtained restriction 108, the default components 706 and 710 are merged for every entry, and the default configuration 714 of the restriction 108 obtained is generated. The exemption setting out 716 of the restriction 108 produced by merging the exemption components 708 and 712 for every entry is generated. After this restriction is drawn, the obtained restriction 108 is used by the GetCryptoPermission method of the JCESecurityManager class 316, and determines the restrictions added to a Cipher instance.

[0054]Next, one embodiment of merge processing is described along with the flow chart of drawing 10 and drawing 11. The next explanation explains using the policy A, B, and C. The policies A and B point out the sources of information (for example, the U.S. exporting method and a local method) of merge. The policy C points out a merge result (for example, obtained restriction 108). As shown in drawing 9, the default components 706 and 710 and the exemption components 708 and 712 are independently merged using separate merge operation. However, note that the same merge processing is used by both merge.

[0055]Now, merge processing begins from selection (804) of the next entry (in this case, the first entry) of the policy A like drawing 10. It is determined whether compare the selected entry with the entry of the policy B, and a corresponding entry exists in the policy B (808). In one embodiment of this invention, this decision is made by comparing the selected algorithm name of an entry and exemption mechanism name with the algorithm name and exemption mechanism name of an entry of the policy B. If the algorithm of the same name as the entry of the policy B and the combination of an exemption mechanism exist, it will become an entry to which the entry corresponds. In this case, restriction of two corresponding entries is compared and it opts for the restriction restrained most (820).

[0056]As an example of this method, the algorithm name of both policies A and B considers the entry in which an exemption mechanism does not exist by RC5. In 64 bits and the maximum repetition number, 12 and the maximum key length of the entry of the policy B consider it as 128 bits, and the maximum repetition number sets [the maximum key length of the entry of the policy A] to 10. In this case, the maximum key length will be 64 bits and, as for the restriction restrained most, the maximum repetition number is set to 10. It opts

for the restriction most restrained for every restriction as shown in this example.
[0057]A new entry is generated by the policy C after the restriction restrained most is decided (824). The same algorithm name as two corresponding entries and an exemption mechanism name exist in this new entry. The restriction for which it opted at the process 820 and which was restrained most exists in this new entry as that restriction. The policy's C generation of a new entry will terminate processing of the entry chosen now. And judgment whether an entry exists in the policy A more is made (828). When an entry exists, processing is returned to the process 804, and the next entry of the policy A is chosen and processed. When an entry does not exist, processing is advanced to the process 832.
[0058]It returns to the process 808, and when it is judged that the entry corresponding to the entry selected in the policy A does not exist in the policy B, judgment whether the entry of a wild card exists in the policy B is made (812). This wild card operates as a container for the combination of all the algorithm name / exemption mechanisms which are not shown in the policy B by showing clearly. When a wild card is not found in the policy B, processing of the selected entry is completed. A new entry is not generated, but processing progresses to the policy C to the process 828, in order to search the next entry of the policy A.
[0059]On the other hand, when it is judged that the entry of a wild card exists in the policy B, the selected restriction of an entry and restriction of the entry of wildcard are compared, and it opts for the restriction restrained most (816). This decision is made by the same method as the explanation mentioned above about the process 820. A new entry is generated by the policy C after the restriction restrained most is decided (824). The same algorithm name as the selected entry and an exemption mechanism name exist in this new entry. The restriction which was decided by the process 816 and which was restrained most exists in this new entry as that restriction. The policy's C generation of a new entry will terminate processing of the entry chosen now. And judgment whether an entry exists in the policy A is made (828). When an entry exists, processing is returned to the process 804, and the next entry of the policy A is chosen and processed. This processing continues until all the entries of the policy A are processed.
[0060]After all the entries of the policy A are processed, it becomes the watch which processes all the entries of the policy B which do not correspond to the entry of the policy A. However, before performing this, it is judged whether the entry of a wild card exists in the policy A (832). When the policy A does not have an entry of a wild card, since they do not become an entry of the addition created by the policy C, the additional entry of the policy B does not need to be processed any more. Thus, when the entry of a wild card does not exist in the policy A, construction of the policy C is completed (836).
[0061]On the other hand, when the entry of a wild card exists in the policy A, processing of the policy B begins from selection of the next entry (in this case, the first entry) of the policy B (840). It is judged whether the selected entry is compared with the entry of the policy C,

and a corresponding entry exists in the policy C (844). In one embodiment of this invention, this decision is made by comparing the algorithm name of the selected entry, an exemption mechanism name, the algorithm name of the entry of the policy C, and an exemption mechanism name. When a corresponding entry is found in the policy C, the selected entry means already having been processed as a part of processing of the entry of the policy A. In this case, processing of the chosen entry is not needed. As a result, processing progresses to the process 856 and the next entry of the policy B is searched.

[0062]On the other hand, when the selected entry supports neither of the entries of the policy C, it opts for the restriction which compared the selected restriction of an entry and restriction of the entry of the wildcard of the policy A, and was restrained most (848). This decision is made by the same method as the explanation mentioned above about the process 820. A new entry is generated by the policy C after opting for the restriction restrained most (852). The same algorithm name as the selected entry and an exemption mechanism name will exist in this new entry. The restriction for which it opted at the process 848 as that restriction and which was restrained most will exist in this new entry. After a new entry is generated by the policy C, processing of the entry selected now is completed. And it is determined whether an entry exists in the policy B further (856). When an entry exists, processing is returned to the process 840, and the next entry of the policy B is chosen and processed. This processing continues until all the entries of the policy B are processed. Processing of all the entries will terminate construction of the policy C (860).

[0063]Explained merge processing is performed by the initializer of the JCESecurity class 314 in one embodiment of this invention. This initializer will be called shortly after the JCESecurity class 314 is called. When initializer is called, initializer merges two or more sets of the law with which initializer was provided, and generates all the sets 108 of restriction. All the sets 108 (it has default configuration and exemption setting out) of this restriction are generated, and this is used after that by the GetCryptoPermission method which determines the restrictions added to a Cipher instance.

[0064]As mentioned above, the GetImpl method of the JCESecurity class 314 takes charge of instantiation of the general related mounting 106, and generates the instance of mounting. As a part of instantiation processing, a GetImpl method performs authenticating processing. In one embodiment of this invention, this authenticating processing becomes the form of the mutual recognition that a GetImpl method attests the general related mounting 106, and the general related mounting 106 attests the framework 102. In order to enable this mutual recognition to produce in one embodiment of this invention, (1) The DESHITARU signature of the JAR file of the general related mounting 106 is carried out, (2) The DESHITARU signature of the JAR file of the framework 102 is carried out, (3) The set of the ambiguous reliance public key (obfuscated trusted public keys) which the JCESecurity class 314 can use for verification of a signature of the JAR file of general

related mounting is embedded, (4) The general related mounting 106 is embedded in the set of the reliance public key used for verification of a signature of the JAR file of a framework.
[0065]This premise can be given and mutual recognition is performed as follows. First, the ambiguous reliance public key embedded in the JCESecurity class 314 is used, and the digital signature of the JAR file of general mounting to which a GetImpl method relates is verified. When this digital signature is verified, a GetImpl method instantiates the general related mounting 106, and the constructor of this general related mounting is called. When a constructor is called, a constructor uses the reliance public key embedded at the general related mounting 106, and verifies the digital signature of the JAR file of a framework. When a constructor determines that the digital signature of the JAR file of a framework is right, a constructor will build the instance of demanded mounting. When a digital signature is not right, a constructor returns an error. The instance of mounting will be built only a right case for both the general related mounting 106 and the framework 102 as shown in this explanation.
[0066]By execution of this verification processing, a GetImpl method trusts an external digital signature verification mechanism. That is, in one embodiment of this invention, the verification of a signature itself does not perform a GetImpl method. On the contrary, a GetImpl method shows an external digital signature verification mechanism the digital signature of the general related mounting 106, and an ambiguous reliance public key, and receives verification. In one embodiment of this invention, an external digital signature verification mechanism turns into a signature mechanism (Signature Mechanism) of Java Runtime. This signature mechanism is a part of total Java environment, and is not a part of framework 102. Therefore, if it sees from the framework 102, this signature mechanism is not the component "trusted." If the result which a signature mechanism can be right and can trust as a result is provided, before trusting it, it is verified in order for the signature mechanism itself to guarantee a lawful thing (that is, the right verifying function is performed).
[0067]In order that it can verify a signature mechanism, at least two digital signatures are embedded into it at the JCESecurity class 314. It turns out that one is verifiable using an ambiguous reliance public key, and, as for another, it turns out that it is unverifiable using an ambiguous reliance public key. These signatures are shown in an order which cannot be predicted in a signature mechanism, and examine the legitimacy. One possible embodiment of processing which examines a signature mechanism is shown in drawing 12.
[0068]Verification processing begins from the determination (904) of the digital signature (digital signature possible [verification] or unverifiable) shown to a signature mechanism as shown in drawing 12. This decision is made by the method which cannot be predicted to be a signature mechanism, and is performed in one embodiment of this invention using random processing. For example, a random number will be generated and, in the case of the

range (it is in agreement with 0) with a random number, one of the signatures will be chosen. In the case of range (it is in agreement with 1) with an another random number, another signature will be chosen. In one embodiment of this invention, a decision of the process 904 is made, even if it takes the before selected signature into consideration. Other signatures are chosen by the process 904 when all before selected signatures are the same signatures. At least one each of two signatures is chosen, and this guarantees examining the legitimacy of a signature mechanism thoroughly.

[0069]After one of the signatures is chosen, it is shown to a signature mechanism for verification of the selected signature and an ambiguous reliance public key (908). Next, a signature mechanism provides the response which shows one of whether the signature was verified or it was not verified. This response is received (912) and accuracy is checked (916). When the signature which the signature mechanism was shown is specifically able to be verified, the response is checked by the index which shows that the signature was verified. When the signature which the signature mechanism was shown is not able to be verified, the response is checked by the index which shows that the signature is not verified. When the response received to the shown signature is not right, it is decided that a signature mechanism will not be lawful (920). In this case, verification processing is completed (924).

[0070]In a right case, the response received to the shown signature on the other hand makes a decision about whether verification processing was performed n times (928). Here, n is arbitrary desirable numbers (for example, 5). When not performing n times, processing is returned to the process 904, a signature is shown to a signature mechanism once again, and a response is examined. When processing is performed n times, processing is advanced to the process 932. When it processes to the process 932, it turns out that the signature mechanism provided the right response to the signature which ****** was shown, respectively (when a response is not right, processing will be completed at the process 924, without resulting to the process 932). Thus, it is decided that a signature mechanism will be lawful (932). In this case, a signature mechanism may be trusted by the GetImpl method which attests the general related mounting 106. Verification of that a signature mechanism is lawful will terminate verification processing (936).

[0071]The result of the above-mentioned processing is that a signature mechanism is shown a verifiable digital signature and unverifiable digital signature in an order [**********]. Even if it is not impossible, it will be very difficult for verification processing "to forge" a right response of an inaccurate signature mechanism by making this presentation order into a prediction impossibility. Therefore, this verification processing provides the efficient method of examining the legitimacy of an external signature mechanism.

[0072]Verification processing was related to the digital signature verification mechanism, and has so far been explained. However, it must be cautious of processing not being limited

to a digital signature verification mechanism. On the contrary, verification processing is generally applied and examines the legitimacy of the arbitrary mechanisms which are not trusted. As long as at least two different information setting exists about the known right response, processing is applied in order to examine the legitimacy of the mechanism which is not trusted. The method generally applied in the mechanisms which are not trusted in which verification processing is arbitrary is shown in the flow chart of drawing 13.

[0073]Verification processing begins from the determination shown to the mechanism in which either of at least two information setting is trusted as shown in drawing 13 (1004). The process 1004 of this determination is performed by the method which cannot be predicted for the mechanism which is not trusted, and is performed in one embodiment of this invention using random processing. For example, a random number will be generated and, in within the limits (it is in agreement with 0) with a random number, the first information setting will be chosen. In within the limits (it is in agreement with 1) of others [number / random], another information setting will be chosen. In one embodiment of this invention, the process 1004 of determination also takes the before selected information setting into consideration. When all before selected selections are the same information setting, other information setting is chosen by the process 1004. This has guaranteed examining thoroughly the legitimacy of the mechanism which each of information setting is chosen once [at least], and is not trusted.

[0074]After one of the information setting is chosen, it shows the mechanism in which the selected information setting is trusted (1008). Next, the mechanism which is not trusted provides the response to the shown information setting. This response is received (1012) and accuracy is checked (1016). Specifically, the right response of each information setting comes to be known. When it is not the right response of the information setting shown the received response, it is determined that the mechanism which is not trusted is not lawful (1020). In this case, verification processing is completed (1024).

[0075]When it is the right response of the information setting shown the received response on the other hand, a decision about whether verification processing was performed n times is made (1028). Here, n is arbitrary desirable numbers (for example, 5). When not performing n times, processing is returned to the process 1004, it shows the mechanism in which information setting is trusted once again, and a response is examined. When processing is performed n times, processing progresses to the process 1032. When it processes to the process 1032, it turns out that the mechanism which is not trusted provided the right response to all the shown information setting, respectively (when a response is not right, processing will be completed at the process 1024, without resulting in the process 1032). Thus, it is determined that the mechanism which is not trusted is lawful (1032). Verification of that the mechanism which is not trusted is lawful will terminate verification processing (1036).

[0076]The result of the above-mentioned processing is shown to the mechanism two information setting not being trusted, in an order [**********]. Even if it is not impossible, it will be very difficult for verification processing "to forge" a right response of the inaccurate mechanism which is not trusted by making this presentation order into a prediction impossibility. Therefore, this verification processing provides the efficient method for examining the legitimacy of the arbitrary mechanisms which are not trusted.
[0077][Outline of hardware] In one embodiment of this invention, this invention is mounted as an instruction set which 1 or two or more processors can perform. This invention can be mounted as some object-oriented programming system containing Sun Microsystems of State Mountain View of California, and the Java (registered trademark) programming system by an Inc. company and which is not limited to this. The block diagram of the hardware of the computer system 1100 in which the embodiment of the invention is mounted is shown in drawing 14. The computer system 1100 contains the processor 1104 which is connected to the bus 1102 used for communication of information or other transmitter styles, and the bus 1102, and processes information. The main memory units 1106, such as random access memory (RAM) or other dynamic storage, are also built into the computer system 1100, and the information and command which it is connected to the bus 1102 and the processor 1104 executes are saved at it. The main memory unit 1106 is used also when the processor 1104 saves a temporary variable or other intermediate information which are used during execution of a command. The computer system 1100 contains the read-only memory (ROM) 1108 or other static storages for which the static information which it is connected to the bus 1102 and the processor 1104 uses, and a command are saved. It is connected to the bus 1102 and the memory storage 1110, such as a magnetic disk or an optical disc, is used for preservation of information and a command.
[0078]It is connected to the display 1112 of a cathode-ray tube (CRT) etc. via the bus 1102, and the computer system 1100 displays information on a computer user. It is connected to the bus 1102 and the input devices 1114 including an alphanumeric character key and other keys send selection of information and a command to the processor 1104. A different user input device from a key has the cursor control 1116, such as a mouse, a trackball, or a cursor arrow key, selection of direction information and a command is sent to the processor 1104, and a motion of the cursor of the display 1112 is controlled. This input device usually has the biaxial flexibility of 2 times of the 1st axis (for example, x) and the 2nd axis (for example, y), and can specify the position within a flat surface.
[0079]According to one embodiment, the computer system 1100 provides the function of this invention according to the processor 1104 which performs 1 stored in the main memory unit 1106, 1 of two or more commands, or two or more sequences. Such a command is read into the main memory unit 1106 from media other than the main memory unit which can read computers, such as the memory storage 1110. Based on the instruction sequence stored in

the main memory unit 1106, the processor 1104 performs the processing step explained here. It can also be used as another embodiment, being able to replace a hard-wired circuit with the software instruction which mounts an invention. A hard-wired circuit can also be used combining the software instruction which mounts an invention. Thus, the embodiment of an invention is not limited to the arbitrary combination of hardware circuitry and software.

[0080]The term of "the medium which a computer can read" currently used here shows the arbitrary media related to providing with a command the processor 1104 used for execution. Although such a medium is not limited to a nonvolatile medium and volatility medium and a transmission medium, there is form of a large number containing them. For example, there are an optical disc or magnetic disks, such as the memory storage 1110, in a nonvolatile medium. There is dynamic storage, such as the main memory unit 1106, in a volatile medium. There are coaxial cables including the wiring which constitutes the bus 1102, copper wire, and an optical fiber in a transmission medium. A transmission medium also becomes forms, such as a sound wave or electromagnetic waves, for example, an electric wave, infrared rays, and a wave generated during optical data communication.

[0081]For example, in a general form of the medium which a computer can read. A floppy (registered trademark) disk, a flexible disk, a hard disk, Magnetic tape or other magnetic media, CD-ROM, other optical media, There is a memory chip of a punch card, a paper streamer, another physical perforation-type medium, RAM, PROM and EPROM, FLASH-EPROM, and others or a cartridge, a subcarrier mentioned later, or a medium which can read other computers.

[0082]Carrying 1, 1 of two or more commands, or two or more sequences to the processor 1104 which executes a command is also included in various kinds of medium which a computer can read. For example, a command is first carried by the magnetic disk on a remote computer. A remote computer loads a command to the dynamic storage, and transmits a command on a telephone wire using a modem. To the computer system 1100, a local modem receives data with a telephone wire, and changes data into an infrared signal using an infrared transmitter. An infrared detector receives the data of an infrared signal and a suitable circuit arranges data on the bus 1102. The bus 1102 carries data to the main memory unit 1106. And the processor 1104 takes out and executes a command. The command which the main memory unit 1106 received is optional to one of the back before the processor 1104 performs, and is saved at the memory storage 1110.

[0083]The computer system 1100 includes again the communication interface 1118 connected to the bus 1102. The data communications of a 2-way are possible for the communication interface 1118, and it is connected also with the network link 1120 for connecting with the local network 1122. For example, the communication interface 1118 can also use an integrated services digital network (ISDN) card or data communication

connection as the modem with which the corresponding telephone wire of a kind is provided. The communication interface 1118 can also use data communication connection as the Local Area Network (LAN) card with which compatible LAN is provided as other examples. A radio link can also be mounted. By such mounting, the communication interface 1118 transmits and receives the electrical and electric equipment, electromagnetism, or lightwave signal which carries the flow of the digital data which is various kinds of information.

[0084]The network link 1120 usually makes possible the data communications to other data stations using 1 or two or more networks. For example, the network link 1120 provides connection to the data station currently managed by the host computer 1124 or Internet Service Provider (ISP) 1126 using the local network 1122. Next, ISP1126 provides data transmission services using the global packet data communication network currently generally called "Internet" 1128. Both the local network 1122 and the Internet 1128 use the electrical signal, the electromagnetic signals, or the lightwave signal which carries the flow of digital data. The signal which uses the signal which uses various networks, the signal on the network link 1120, and the communication interface 1118 has a form of the typical subcarrier which carries information. These signals are carried to the computer system 1100, and carry digital data from the computer system 1100.

[0085]The computer system 1100 can transmit a message using a network, the network link 1120, and the communication interface 1118, and can receive data including a program code. In the Internet, when the code of an application program is required, the server 1130 may be able to transmit it using the Internet 1128, ISP1126, the local network 1122, and the communication interface 1118. When the received code receives a code, it can be executed by the processor 1104, or can be saved at the memory storage 1110 or other nonvolatile storages, and can be executed later. By this method, the computer system 1100 can acquire application codes in the form of a subcarrier.

[0086]At present, although the invention is explained based on the special embodiment, it is not limited to it. Various change can be made by a person skilled in the art using the profits of this indication, without deviating from the pneuma of an invention. Therefore, this invention is not restricted to the specific embodiment currently used in order to describe this invention, and only by being based on a patent generic claim, it is limited.

[0087]

[Effect of the Invention]According to this invention, a framework makes it a positive thing to impose required restrictions on the service provided for application according to the demanded dynamic construction of mounting like the above.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1]It is a block diagram showing the whole system concerning one embodiment of this invention.

[Drawing 2]It is a flow chart showing general operation of the whole system of drawing 1.

[Drawing 3]It is a detailed block diagram showing one embodiment of this invention.

[Drawing 4]It is a detailed block diagram showing one embodiment of this invention.

[Drawing 5]It is a flow chart showing operation of the embodiment of drawing 3 and drawing 4.

[Drawing 6]It is a flow chart showing operation of the embodiment of drawing 3 and drawing 4.

[Drawing 7]It is a figure showing the example of the combination of restrictions including default configuration and exemption setting out.

[Drawing 8]It is a flow chart showing operation of one embodiment of the GetCryptoPermission method of JCESecurityManager object classes.

[Drawing 9]It is a flow chart of one embodiment of this invention showing the outline of the processing merged into one set of restriction of two or more sets of a rule.

[Drawing 10]It is a flow chart of one embodiment of this invention showing how to merge two or more sets of a rule into one set of restriction.

[Drawing 11]It is a flow chart of one embodiment of this invention showing how to merge two or more sets of a rule into one set of restriction.

[Drawing 12]It is a flow chart of one embodiment of this invention showing how to examine the legitimacy of the digital signature verification mechanism which is not trusted.

[Drawing 13]It is a flow chart of one embodiment of this invention showing how to examine the legitimacy of the arbitrary mechanisms which are not trusted.

[Drawing 14]It is a hardware block diagram of the computer system which this invention mounts.

---

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2001－216043
（P2001－216043A）
(43)公開日　平成13年８月10日(2001.8.10)

(51)Int.Cl.⁷		識別記号	FI		テーマコード(参考)
G06F	1/00		G09C	1/00	640B
G09C	1/00	640	G06F	9/06	660G

審査請求　未請求　請求項の数42　ＯＬ　（全 22 頁）

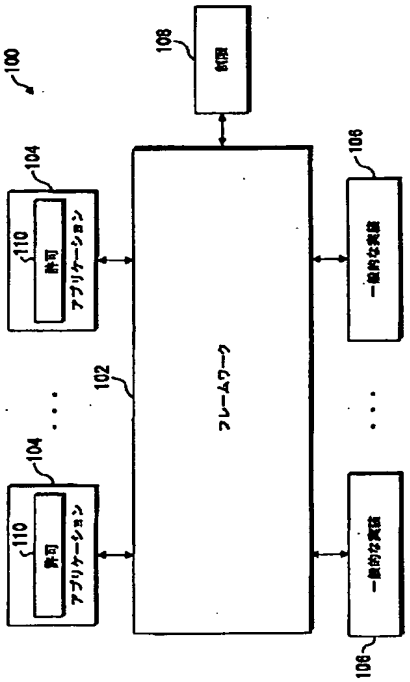
(21)出願番号　特願2000－354265(P2000－354265)
(22)出願日　平成12年11月21日(2000.11.21)
(31)優先権主張番号　６０／１６６８７１
(32)優先日　平成11年11月22日(1999.11.22)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　６０／１７４６３０
(32)優先日　平成12年１月５日(2000.1.5)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０９／４８３２４６
(32)優先日　平成12年１月14日(2000.1.14)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　398014344
サン・マイクロシステムズ・インコーポレーテッド
アメリカ合衆国　94303　カリフォルニア州　パロアルト　エムエス　ピーエーエルオー　１－521　サンアントニオロード　901
(72)発明者　シャロン・エス・リウ
アメリカ合衆国　95014　カリフォルニア州　クーパチノ　デオダラ　ドレイブ　10558
(74)代理人　100098039
弁理士　遠藤　恭

最終頁に続く

(54)【発明の名称】　サービスの実装に課する制約を決定するための機構

(57)【要約】　（修正有）
【解決手段】アプリケーションが特定サービスのための実装を求めるとき、アプリケーションはフレームワークに要求を出す。フレームワークは、この要求を受け取り、これに対応して、制約が存在する場合は、どの制約が要求された実装に課す必要のあるものかを決定する。この制約は、実装を要求しているアプリケーションに与えられた許可があるか否かを判断することにより決定され、この許可がある場合、前記実装に課する制約のセットを導くためにこの許可を処理する。この許可は、導かれる制約のセットが最も小さい制約度合いとなるよう処理される。
【効果】要求された実装の動的な構築により、フレームワークは、必要な制約がアプリケーションに提供されるサービスに課されることを確実なものとする。



【特許請求の範囲】
【請求項１】　アプリケーションにより特定サービスの実装が要求されるシステムにおける、前記実装に課す制約を決定する方法であって、
前記実装を要求する前記アプリケーションに与えられている許可があるか否かを判断し、
前記アプリケーションに与えられている少なくとも１つの許可があるとの判断に応じて、前記実装に課す制約のセットを導くために、前記許可を処理することを備えた方法。
【請求項２】　前記許可は、前記制約のセットが最も小さい制約度合いとなるよう処理されるものである請求項１に記載の方法。
【請求項３】　前記アプリケーションに与えられている許可がないとの判断に応じて、初期制限のセットにアクセスし、
前記初期制限に基づいて、前記制約を導くことを更に備えた請求項１に記載の方法。
【請求項４】　前記初期制限が、複数の法政策をマージし、そこから最も制約的な制限を抽出することにより、導かれるものである請求項３に記載の方法。
【請求項５】　前記許可の前記処理が、
前記許可が、全てを包含する許可であるか否かを判断し、
前記許可が全てを包含する許可であるとの判断に応じて、前記実装が制限のないものであることの指標を用意することを備えた請求項１に記載の方法。
【請求項６】　前記許可の前記処理が、
前記許可が、免除機構が実装されることを必要とするか否かを判断し、
前記許可が免除機構が実装されることを必要としないとの判断に応じて、前記許可に基づき前記制約のセットを導くことを備えた請求項１に記載の方法。
【請求項７】　前記制約を導くことが、
前記許可がパラメータのセットを規定しているか否かを判断し、
前記許可がパラメータのセットを規定しているとの判断に応じて、前記パラメータに基づき前記制約のセットを導くことを備えた請求項６に記載の方法。
【請求項８】　前記制約を導くことが、
前記許可がパラメータのセットを規定しているか否かを判断し、
前記許可がパラメータのセットを規定していないとの判断に応じて、前記実装が制限のないものであることの指標を用意することを備えた請求項６に記載の方法。
【請求項９】　前記許可の前記処理が、
前記許可が、特定の免除機構が実装されることを必要とするか否かを判断し、
前記許可が特定の免除機構が実装されることを必要とするとの判断に応じて、免除制限のセットをアクセスし、前記制約を導くために、前記許可と前記免除制限とを調整することを備えた請求項１に記載の方法。
【請求項１０】　前記許可と前記免除制限との前記調整が、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するか否かを判断し、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するとの判断に応じて、前記免除制限に基づいて前記制約を導くことを備えた請求項９に記載の方法。
【請求項１１】　前記許可と前記免除制限との前記調整が、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するか否かを判断し、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可しないとの判断に応じて、初期制限のセットにアクセスし、
前記初期制限に基づいて前記制約を導くことを備えた請求項９に記載の方法。
【請求項１２】　前記免除制限が、複数の法政策をマージし、そこから最も制約的な制限を抽出することにより、導かれるものである請求項９に記載の方法。
【請求項１３】　前記アプリケーションに与えられている許可があるか否かの前記判断が、前記実装を要求したアプリケーションが何れであるかを判断するためにコール・スタックを精査することを備えた請求項１に記載の方法。
【請求項１４】　前記アプリケーションに与えられている許可があるか否かの前記判断が、前記アプリケーションを認証することを更に備えた請求項１３に記載の方法。
【請求項１５】　アプリケーションにより特定サービスの実装が要求されるシステムにおける、前記実装に課す制約を決定する装置であって、
前記実装を要求する前記アプリケーションに与えられている許可があるか否かを判断する機構と、
前記アプリケーションに与えられている少なくとも１つの許可があるとの判断に応じて、前記実装に課す制約のセットを導くために、前記許可を処理する機構と、を備えた装置。
【請求項１６】　前記許可は、前記制約のセットが最も小さい制約度合いとなるよう処理されるものである請求項１５に記載の装置。
【請求項１７】　前記アプリケーションに与えられている許可がないとの判断に応じて、初期制限のセットにアクセスする機構と、
前記初期制限に基づいて、前記制約を導くする機構と、を更に備えた請求項１５に記載の装置。
【請求項１８】　前記初期制限が、複数の法政策をマージし、そこから最も制約的な制限を抽出することによ

り、導かれるものである請求項１７に記載の装置。
【請求項１９】　前記許可を処理する前記機構が、
前記許可が、全てを包含する許可であるか否かを判断する機構と、
前記許可が全てを包含する許可であるとの判断に応じて、前記実装が制限のないものであることの指標を用意する機構と、を備えた請求項１５に記載の装置。
【請求項２０】　前記許可を処理する前記機構が、
前記許可が、免除機構が実装されることを必要とするか否かを判断する機構と、
前記許可が免除機構が実装されることを必要としないとの判断に応じて、前記許可に基づき前記制約のセットを導く機構と、を備えた請求項１５に記載の装置。
【請求項２１】　前記制約を導く前記機構が、
前記許可がパラメータのセットを規定しているか否かを判断する機構と、
前記許可がパラメータのセットを規定しているとの判断に応じて、前記パラメータに基づき前記制約のセットを導く機構と、を備えた請求項２０に記載の装置。
【請求項２２】　前記制約を導く前記機構が、
前記許可がパラメータのセットを規定しているか否かを判断する機構と、
前記許可がパラメータのセットを規定していないとの判断に応じて、前記実装が制限のないものであることの指標を用意する機構と、を備えた請求項２０に記載の装置。
【請求項２３】　前記許可を処理する前記機構が、
前記許可が、特定の免除機構が実装されることを必要とするか否かを判断する機構と、
前記許可が特定の免除機構が実装されることを必要とするとの判断に応じて、免除制限のセットをアクセスする機構と、
前記制約を導くために、前記許可と前記免除制限とを調整する機構と、を備えた請求項１５に記載の装置。
【請求項２４】　前記許可と前記免除制限とを調整する前記機構が、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するか否かを判断する機構と、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するとの判断に応じて、前記免除制限に基づいて前記制約を導く機構と、を備えた請求項２３に記載の装置。
【請求項２５】　前記許可と前記免除制限とを調整する前記機構が、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するか否かを判断する機構と、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可しないとの判断に応じて、初期制限のセットにアクセスする機構と、
前記初期制限に基づいて前記制約を導く機構と、を備えた請求項２３に記載の装置。
【請求項２６】　前記免除制限が、複数の法政策をマージし、そこから最も制約的な制限を抽出することにより、導かれるものである請求項２３に記載の装置。
【請求項２７】　前記アプリケーションに与えられている許可があるか否かを判断する前記機構が、前記実装を要求したアプリケーションが何れであるかを判断するためにコール・スタックを精査する機構を備えた請求項１５に記載の装置。
【請求項２８】　前記アプリケーションに与えられている許可があるか否かを判断する前記機構が、前記アプリケーションを認証する機構を更に備えた請求項２７に記載の装置。
【請求項２９】　命令を格納したコンピュータ読み取り可能な媒体であって、前記命令が、１又は複数のプロセッサにより実行されたときに、該１又は複数のプロセッサを、アプリケーションにより要求された特定サービスの実装に課す制約を決定するように動作させるものにおいて、前記コンピュータ読み取り可能な媒体が、
前記実装を要求する前記アプリケーションに与えられている許可があるか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記アプリケーションに与えられている少なくとも１つの許可があるとの判断に応じて、前記実装に課す制約のセットを導くために、前記許可を処理するよう、１又は複数のプロセッサを動作させる命令と、を備えたコンピュータ読み取り可能な媒体。
【請求項３０】　前記許可は、前記制約のセットが最も小さい制約度合いとなるよう処理されるものである請求項２９に記載のコンピュータ読み取り可能な媒体。
【請求項３１】　前記アプリケーションに与えられている許可がないとの判断に応じて、初期制限のセットにアクセスするよう、１又は複数のプロセッサを動作させる命令と、
前記初期制限に基づいて、前記制約を導くするよう、１又は複数のプロセッサを動作させる命令と、を更に備えた請求項２９に記載のコンピュータ読み取り可能な媒体。
【請求項３２】　前記初期制限が、複数の法政策をマージし、そこから最も制約的な制限を抽出することにより、導かれるものである請求項３１に記載のコンピュータ読み取り可能な媒体。
【請求項３３】　前記許可を処理するよう、１又は複数のプロセッサを動作させる前記命令が、
前記許可が、全てを包含する許可であるか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記許可が全てを包含する許可であるとの判断に応じて、前記実装が制限のないものであることの指標を用意するよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項２９に記載のコンピュータ読み取り

可能な媒体。
【請求項３４】　前記許可を処理するよう、１又は複数のプロセッサを動作させる前記命令が、
前記許可が、免除機構が実装されることを必要とするか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記許可が免除機構が実装されることを必要としないとの判断に応じて、前記許可に基づき前記制約のセットを導くよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項２９に記載のコンピュータ読み取り可能な媒体。
【請求項３５】　前記制約を導くよう、１又は複数のプロセッサを動作させる前記命令が、
前記許可がパラメータのセットを規定しているか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記許可がパラメータのセットを規定しているとの判断に応じて、前記パラメータに基づき前記制約のセットを導くよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項３４に記載のコンピュータ読み取り可能な媒体。
【請求項３６】　前記制約を導くよう、１又は複数のプロセッサを動作させる前記命令が、
前記許可がパラメータのセットを規定しているか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記許可がパラメータのセットを規定していないとの判断に応じて、前記実装が制限のないものであることの指標を用意するよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項３４に記載のコンピュータ読み取り可能な媒体。
【請求項３７】　前記許可を処理するよう、１又は複数のプロセッサを動作させる前記命令が、
前記許可が、特定の免除機構が実装されることを必要とするか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記許可が特定の免除機構が実装されることを必要とするとの判断に応じて、免除制限のセットをアクセスするよう、１又は複数のプロセッサを動作させる命令と、
前記制約を導くために、前記許可と前記免除制限とを調整するよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項２９に記載のコンピュータ読み取り可能な媒体。
【請求項３８】　前記許可と前記免除制限とを調整するよう、１又は複数のプロセッサを動作させる前記命令が、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するとの判断に応じて、前記免除制限に基づいて前記制約を導くよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項３７に記載のコンピュータ読み取り可能な媒体。
【請求項３９】　前記許可と前記免除制限とを調整するよう、１又は複数のプロセッサを動作させる前記命令が、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可するか否かを判断するよう、１又は複数のプロセッサを動作させる命令と、
前記免除制限が、前記実装に前記特定の免除機構が実装されることを許可しないとの判断に応じて、初期制限のセットにアクセスするよう、１又は複数のプロセッサを動作させる命令と、
前記初期制限に基づいて前記制約を導くよう、１又は複数のプロセッサを動作させる命令と、を備えた請求項３７に記載のコンピュータ読み取り可能な媒体。
【請求項４０】　前記免除制限が、複数の法政策をマージし、そこから最も制約的な制限を抽出することにより、導かれるものである請求項３７に記載のコンピュータ読み取り可能な媒体。
【請求項４１】　前記アプリケーションに与えられている許可があるか否かを判断するよう、１又は複数のプロセッサを動作させる前記命令が、前記実装を要求したアプリケーションが何れであるかを判断するためにコール・スタックを精査するよう、１又は複数のプロセッサを動作させる命令を備えた請求項２９に記載のコンピュータ読み取り可能な媒体。
【請求項４２】　前記アプリケーションに与えられている許可があるか否かを判断するよう、１又は複数のプロセッサを動作させる前記命令が、前記アプリケーションを認証するよう、１又は複数のプロセッサを動作させる命令を更に備えた請求項４１に記載のコンピュータ読み取り可能な媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、コンピュータシステムに関し、特に、アプリケーションにより要求されたサービスの実装に課する制約を決定するための機構に関する。
【０００２】
【従来の技術】長年、米国商務省は、データ暗号化アルゴリズムを含むコンピュータプログラムまたはアプリケーションの輸出を規制し、場合によっては禁止してきた。現在原則として、一定ビット数以上の暗号鍵を用いる暗号化アルゴリズムを使用しているコンピュータプログラムは輸出できない（指定可能な鍵の長さはアルゴリズム特有）。この規則には例外もある。例外の１つは、免除機構が採用されている場合で、鍵の長さ、すなわちプログラムの暗号強度を、場合によっては増加できる。

免除機構の例には、キー・エスクロー(key escrow)、キー・リカバリー(key recovery)およびキー・ウィークニング(key weakening)がある。また、プログラムの種類によっては鍵の長さを大きくできる。たとえば、現在の規制は、医療機関および金融機関向けのアプリケーションが鍵の長さを大きくして、アプリケーションの安全性を高める（高度の機密データの保護に使用する）ことを許容している。他のアプリケーションよりも許容範囲の大きい恵まれたアプリケーションがある一方で、暗号化アプリケーション全てに輸出規制が必要になる。

【０００３】

【発明が解決しようとする課題】これらの規制は、暗号化アルゴリズムを直接使用しているプログラムに対して適用されるだけではなく、暗号化アルゴリズムを直接使用しているプログラムに対してインターフェイスを持っているプログラムに対しても適用される。プログラムには、いろいろなプログラム間の相互作用を円滑に行うためのインフラを提供する「フレームワーク」プログラムが含まれる。フレームワーク自体はどのような暗号化アルゴリズムも実装しないが、暗号化アルゴリズムを実装している１つまたは複数のプログラムが、フレームワークに対してインターフェイスし、又はフレームワークに「プラグイン」することを許容できる。このようなフレームワークの例の１つがカルフォルニア、パロアルトのサンマイクロシステムズ社製のJava PlatformのJava Cryptography Extensionである。フレームワークが暗号機構をフレームワークに「プラグイン」することを許容する場合には、フレームワーク自体に輸出規制が必要になる。このことは、輸出ができるようにするには、フレームワークにプラグインされている暗号実装にかかわらず、全ての輸出規制が守られていることをフレームワークが保証する必要があることを意味している。この保証を行うためには、フレームワークがいずれかの機構によって暗号実装を制約する必要がある。

【０００４】

【課題を解決するための手段】本発明に従えば、サービスに制約を課すカスタム化された実装を動的に構築する機構が提供される。本発明の目的のために、サービスは広義に定義され、暗号化/復号化機能などを含む（しかし、これに限定されない）アプリケーションによって要求又は該アプリケーションに対して提供される一切の機能を包含する。本発明の一実施形態において、発明は、アプリケーション、特定サービスの一般的な実装およびフレームワークを備えるシステム内で実現されている。

【０００５】フレームワークは、アプリケーションから特定サービスの実装、たとえば特定暗号化アルゴリズムの実装の要求を受け取る。これに対応して、フレームワークは、制約が存在する場合は、要求された実装に課す必要のある制約を決定する。一実施形態において、フレームワークは、前記アプリケーションに与えられた許可があるか否かを判断することにより、この制約を決定し、もしこの許可がある場合、前記実装に課する制約のセットを導くためにこの許可を処理する。一実施形態において、この許可は、導かれる制約のセットが最も小さい制約度合いとなるよう処理される。この制約が決定されると、フレームワークは、要求された実装を動的に構築する。一実施形態において、要求された実装は、それが前記サービスの一般的な実装、前記制約、及び前記一般的な実装に制約を課する施行ロジックを組み込むように構築される。要求された実装は、前記アプリケーションを対象に構築されるので、そのアプリケーション用にカスタマイズされたものとなる。従って、この実装は、カスタム化された実装と呼ばれる。

【０００６】カスタム化された実装が動的に構築された後、フレームワークは、カスタム化された実装をアプリケーションに提供する。その後、アプリケーションはサービスのためにカスタム化された実装を直接呼び出す。カスタム化された実装には、制約とそれを課するための施行ロジックが組み込まれているので、アプリケーションは更にフレームワークと相互に作用する必要がない。カスタム化された実装自体がサービスを提供し、確実に制約が加えられることになる。このようにカスタム化された実装の動的な構築によって、フレームワークが、アプリケーションに対して提供されるサービスに必要な制約を加えることができる。

【０００７】

【発明の実施の形態】図１に、本発明の実施形態の１つが実現されているシステム１００のブロック図を示す。このシステム１００には、１または複数のアプリケーション１０４、１または複数の一般的な実装１０６、指定された制限のセット１０８および各種のコンポーネント間の相互作用を円滑に行うためのフレームワーク１０２が含まれる。アプリケーション１０４が、サービスの実装をフレームワーク１０２に要求し、受け取る。ここで、アプリケーション１０４は、各種のアプリケーション又はプログラムでよく、Ｊａｖａアプレット、Ｊａｖａアプリケーションおよびネイティブにコンパイルされたアプリケーション（これらに限定されない）などを含む。本発明の目的のために、「サービス」という用語は広義に定義され、暗号化/復号化機能などを含む（しかし、これに限定されない）アプリケーションによって要求又は該アプリケーションに対して提供される一切の機能を包含する。

【０００８】アプリケーション１０４は、実装をフレームワーク１０２に要求する場合、それが望む実装のためのサービスの種類を指定する。たとえば、アプリケーション１０４は「Blowfish」暗号化アルゴリズムの実装を要求できる。これに対応して、フレームワーク１０２は、要求を出しているアプリケーション１０４用にカスタム化された、要求サービスの実装をアプリケーション

１０４に提供する。フレームワーク１０２によって提供されるカスタム化された実装には、フレームワークが提供するサービスの制約が含まれる。後述するように、これらの制約は、指定された制限１０８のセット、及びもしあれば、要求を出しているアプリケーション１０４に認められた許可１１０に基づいて決定される。

【０００９】一般的な実装１０６は、フレームワーク１０２に「プラグイン」されるか、インターフェイスされるサービスの実装を表している。一般的な実装１０６の各々は、特定種類のサービスを実現する。たとえば、１つの一般的な実装が「Blowfish」暗号化アルゴリズムを実装し、同時に、もう１つの実装がＤＥＳ暗号化アルゴリズムを実装する。一般的な実装１０６の各々は、制約されない。すなわち、制限１０８、または許可１１０が存在しても、一般的な実装１０６自体は制約によって妨げられない。これによって、一般的な実装１０６が暗号化アルゴリズムの実装の場合、暗号化アルゴリズムを完全な強度に設定できる。次に説明するように、適当な制約がアプリケーション１０４に提供されるサービスに加えられることを保証するのはフレームワーク１０２であって、一般的な実装１０６ではない。

【００１０】システム１００において、フレームワーク１０２は、システム１００の動作全体を調整するコンポーネントである。フレームワーク１０２の一般的な動作を示す流れ図を、図２に示す。図２に示しているように、フレームワーク１０２は、アプリケーション１０４から特定種類のサービスの実装（たとえば、Blowfish暗号化アルゴリズムの実装）の要求を受け取ることによって動作する（２０２）。これに対応して、制約が存在する場合に、フレームワーク１０２は要求された実装に必要な制約を判断する（２０４）。本発明の一実施形態において、フレームワーク１０２は、指定された制限１０８を、もしあれば、要求を出しているアプリケーション１０４に認められた許可１１０と調整することによって制約を判断する。そして、本発明の一実施形態において、フレームワーク１０２は可能な限り、最低限の制約を加えようとする。言い換えると、フレームワーク１０２は、許可１１０と制限１０８を考慮して可能な限り寛大であるように試みる。

【００１１】制約が決まると、フレームワーク１０２は、要求された実装を動的に構築する（２０６）。本発明の一実施形態において、要求された実装は、要求されたサービスの種類（たとえば、Blowfish暗号化アルゴリズムを実現する一般的な実装１０６）を実装する、関連する一般的な実装１０６を見つけることによって構成される。これが見つかると、この関連する一般的な実装１０６は、前で決定された制約と共に、要求された実装に組み込まれる。また、施行ロジックのセットもこの要求された実装に組み込まれる。この施行ロジックは、この制約が、関連する一般的な実装１０６に課されることを保証する。従って、関連する一般的な実装１０６自体には制約が付いていないにもかかわらず、施行ロジックが、関連する一般的な実装１０６に適用される適当な制約となる。関連する一般的な実装、ここに組み込まれた制約、および施行ロジックを使用して、要求された実装の構築が完了する。要求された実装が、要求しているアプリケーション１０４用に特別に構築、すなわち、要求しているアプリケーション特有の制約を組み込むので、要求された実装は、要求しているアプリケーション１０４用にカスタム化された実装として見ることができる。

【００１２】カスタム化された実装が構築されると、要求しているアプリケーション１０４にこの実装が渡される（２０８）。その後、アプリケーション１０４がカスタム化された実装にサービスを直接要求する。カスタム化された実装に、制約と制約を課すための施行ロジックが組み込まれているので、アプリケーション１０４が更にフレームワーク１０２と相互に作用する必要がなくなる。カスタム化された実装自体がサービスを提供し、確実に制約がサービスに加えられるようになる。この方法のようにカスタム化された実装を動的に構築することによって、フレームワーク１０２が、アプリケーション１０４に提供されるサービスに必要な制約を確実に加える。

【００１３】前述の説明は、本発明の一般的な概要を提供している。図３で、本発明の一実施形態について詳細に説明する。以下の説明で、本発明は、要求され、提供されるサービスが暗号サービスであるオブジェクト指向の実装に関連して説明されている。これが説明の目的のためだけに使用されていることに注意されたい。本発明は説明の範囲に限定されない。適切に言えば、本発明は、あらゆる種類のプログラミング環境と、制約が加えられる必要のあるあらゆる種類のサービスに対して、一般的に適用される。

【００１４】図３及び図４に、フレームワーク１０２の詳細を示す。図に示されているように、フレームワーク１０２は、アプリケーション・プログラミング・インターフェイス（ＡＰＩ）３０２、サービスプロバイダ・インターフェイス（ＳＰＩ）３０４およびコア３２０を備える。ＡＰＩ３０２は、アプリケーション１０４が直接呼び出すことができるリソースを表している。本発明の一実施形態において、ＡＰＩ３０２はCipherオブジェクトクラス３０６とExemptionMechanismオブジェクトクラス３０８を備える。他のメソッドに混じって、Cipherオブジェクトクラス３０６はGetInstanceメソッドとInitメソッドを備える。GetInstanceメソッドは、アプリケーションがサービスの実装を要求する場合に、アプリケーション１０４によって呼び出されるメソッドである。このメソッド呼び出しに対応して、Cipherオブジェクトクラス３０６のインスタンスが構築され、呼び出しているアプリケーション１０４に返される。返されたCipher

インスタンスは、呼び出しているアプリケーション用にカスタム化され、制約と、Cipherインスタンスが提供できるサービスにこの制約を加えるための施行ロジックを含んでいる。Cipherインスタンスが返されると、Cipherインスタンスのメソッドがアプリケーション１０４によって直接呼び出される。呼び出しているアプリケーション１０４によって呼び出される必要のあるメソッドの１つがInitメソッドである。このメソッドは、Cipherインスタンスを初期化し、Cipherインスタンスが動作できるようにする。また、InitメソッドはCipherインスタンスに制約を加えるための施行ロジックとして動作する。GetInstanceメソッドとInitメソッドについては詳細に後述する。

【００１５】すでに説明したように、１または複数の免除機構（キー・エスクロー、キー・リカバリーまたはキー・ウィークニング）が実装されている場合、場合によっては、暗号強度を強化した（たとえば、鍵長を長くした）暗号化アルゴリズムを実現できる。免除機構が実装されている場合に、ExemptionMechanismオブジェクトクラス３０８が動作する。このクラスは、呼び出し可能な複数のメソッドを提供する。これらのメソッドは、特定の免除機構の機能を呼び出す（たとえば、免除機構がキー・リカバリーである場合、キー・リカバリーブロックを生成する）ためと、必要な動作が実行されたかどうか（たとえば、キー・リカバリーブロックが生成されたかどうか）を判断するために呼び出される。ＡＰＩ３０２のオブジェクトクラス３０６、３０８については、後で詳細に説明する。

【００１６】ＳＰＩ３０４は、サービスプロバイダに必要なインターフェイスを提供して、サービスプロバイダのサービス実装をフレームワーク１０２にプラグインする。本発明の一実施形態において、ＳＰＩ３０４は、各ＡＰＩ３０２オブジェクトクラスに対応するＳＰＩ３０４オブジェクトクラスを備える。すなわち、ＡＰＩ３０２のCipherオブジェクトクラス３０６に対して対応するＳＰＩ３０４のCipherSpiオブジェクトクラス３１０が存在する。そして、ＡＰＩ３０２のExemptionMechanismオブジェクトクラス３０８に対して対応するＳＰＩ３０４のExemptionMechanismSpiオブジェクトクラス３１２が存在する。この１対１の対応は、ＡＰＩクラス３０６、３０８のメソッドをＳＰＩクラス３１０、３１２のメソッドにマッピングすることを容易にする。この重要性は詳細に後述される。ＳＰＩオブジェクトクラス３１０と３１２は抽象オブジェクトクラスであり、クラスによって実装されなければならないメソッドを示していると同時に、オブジェクトクラス自体がこれらのメソッドの実装を何も提供しないことを意味している。実装の提供は、サービスプロバイダが担当する。サービスの実装１０６を提供するには、サービスプロバイダがＳＰＩ３０４のオブジェクトクラスの１つをサブクラス化し、そのサブクラスにＳＰＩクラスの定義されたメソッドの全てのために実装を用意する。このようにして、図３に示されている一般的な実装１０６は、ＳＰＩ３０４のオブジェクトクラス３１０と３１２のサブクラスになる。一般的な実装１０６の各々は、異なった種類のサービスも実装でき（たとえば、Blowfish暗号化アルゴリズムを実装し、同時に、ＤＥＳ暗号化アルゴリズムを実装し、そして同時に、キー・リカバリー免除機構を実装できる）、一般的な実装１０６の各々が何の制約も受けずに実現できる。これよって、一般的な実装１０６を最大限の強度の実装（たとえば、暗号鍵の長さを無制限）にすることもできる。フレームワーク１０２のコア３２０は、JCESecurityオブジェクトクラス３１４とJCESecurityManagerオブジェクトクラス３１６を備える。本発明の一実施形態において、これらのオブジェクトクラス３１４と３１６はパッケージプライベートであり、アプリケーション１０４が直接アクセスできない。図３に示されているように、JCESecurityクラスはGetImplメソッドを備え、JCESecurityManagerクラスはGetCryptoPermissionメソッドを備える。これらのメソッドは、Cipherクラス３０６のGetInstanceメソッドの呼び出しの結果として呼び出され、協働して、カスタム化された実装の動的な構築に必要な多数の作業を行う。これらの方法によって実行される機能は、全システムの内容が分かれば、よく理解できる。従って、次に、発明の全てを円滑に理解するために、図５及び図６の流れ図でシステムの全ての動作を説明する。

【００１７】アプリケーション１０４は、それが特定の暗号サービスの実装を必要としている場合に、Cipherオブジェクトクラス３０６のGetInstanceメソッドを呼び出すことよって実装の要求を出す。この呼び出しでは、アプリケーションが実装を要求しているサービスの種類を指定する。本発明の一実施形態において、サービスの種類は、たとえばBlowfishのような暗号化アルゴリズム名になる。Cipherクラス３０６はこの要求を受け取り（４０４）、GetInstanceメソッドの機能を呼び出す。これに対応して、GetInstanceメソッドは、JCESecurityクラス３１４のGetImplメソッドを呼び出す。GetImplメソッドは、複数の重要な機能を実行する。このメソッドは、まず、要求された種類のサービスを実装している一般的な実装１０６が使用可能かどうか判断する（４０８）。たとえば、一般的な実装１０６のいずれかがBlowfish暗号化アルゴリズムを実装しているかどうか判断する。適当な一般的な実装１０６が見つからなかった場合、GetInstanceメソッドにエラーメッセージを返し（４１２）、次に、呼び出しているアプリケーション１０４にエラーメッセージを返す。逆に、要求されたサービスを実装している一般的な実装１０６が見つかった場合、GetImplメソッドは、見つかった一般的な実装が認証されているかどうか続けて判断する（４１６）。この

認証を行う方法については詳細に後述するが、ここでは、認証が、デジタル署名検証機構を使用して行われる、とだけ説明しておく。

【００１８】GetImplメソッドがその一般的な実装が認証されていないと判断した場合、要求されているサービスが行える一般的な実装１０６が他に存在しないかどうか判断する（４２０）。他に一般的な実装１０６が存在しない場合、GetImplメソッドはGetInstanceメソッドにエラーメッセージを返し（４２４）、次に、呼び出しているアプリケーション１０４にエラーメッセージを返す。要求されているサービスが行える一般的な実装１０６が他に存在する場合、GetImplメソッドは、工程４１６に戻って、新しい一般的な実装が認証されているかどうか判断する。この処理が、認証された実装が見つかるか、要求されたサービスが行える認証された一般的な実装１０６が存在しないと判断されるまで続く。

【００１９】要求されたサービスの認証された一般的な実装１０６（この実装は関連する実装と呼ばれることになる）が見つかった場合、GetImplメソッドは関連する実装をインスタンス化して（４２８）実装のインスタンス（すなわちCipherＳＰＩインスタンス）を生成する。その後、GetImplメソッドが何らかの制限を実装のインスタンスに加える必要があるかどうか判断する（４３２）。本発明の一実施形態において、この判断は、フレームワーク１０２が国内的な動作または世界的な動作のために設定されているかどうか判断することによって行われる。フレームワークが国内での使用に限られるように設定されている場合、輸出規制が適用されないので、制約を加える必要はない。一方、フレームワーク１０２が世界的な動作を行うように設定されている場合、制限の可能性が考慮される。

【００２０】実装のインスタンスに加えられる制約を判断するために（４３６）、GetImplメソッドはJCESecurityManagerクラス３１６のGetCryptoPermissionメソッドを呼び出す。GetCryptoPermissionメソッドの重要な機能は、指定された制限１０８と、もしあれば、呼び出しているアプリケーション１０４に認められた許可１１０とを調整し、制約のセットを導くことである。この制約のセットがGetCryptoPermissionメソッドによってGetImplメソッドに返される。そして、本発明の一実施形態において、この制約のセットには、要求された暗号化アルゴリズムの名前、（もし存在するとすれば）課せられる必要のある免除機構の名前、使用される最大鍵長や暗号の最大実行繰り返し数（ＲＣ５などのアルゴリズムに必要）などの幾つかの暗号パラメータが含まれる。GetImplメソッドは、この制約を受け取ると、免除機構がこの制約内で指定されているかどうか判断する（４４０）。免除機構が制約内で指定されていない場合、GetImplメソッドは、工程４４８に進む。

【００２１】しかし、免除機構が指定されている場合、GetImplメソッドは指定された免除機構のインスタンスを続けて生成する。本発明の一実施形態において、これは、ExemptionMechanismクラス３０８のGetInstanceメソッドを呼び出し、免除機構の名前を伝えることによって達成される。この呼び出しに対してExemptionMechanismクラス３０８のGetInstanceメソッドは、JCESecurityクラス３１４のGetImplメソッドを呼び出す（この呼び出しは、GetImplメソッドの第２の呼び出しになることに注意されたい）。これに対応して、GetImplメソッドは指定された免除機構を実装している有効な一般的な実装１０６を検索し、その一般的な実装１０６をインスタンス化して（４４４）、ExemptionMechanismSpiインスタンスを生成する。その後、GetImplメソッドはExemptionMechanismSpiインスタンスをExemptionMechanismクラス３０８のGetInstanceメソッドに返す（これは、GetImplメソッドの第２の呼び出しからの復帰である）。

【００２２】次に、ExemptionMechanismクラス３０８のGetInstanceメソッドは、ExemptionMechanismクラス３０８のコンストラクタを呼び出し、GetImplメソッドから戻されたExemptionMechanismSpiインスタンスをコンストラクタに渡す。呼び出された場合に、コンストラクタはExemptionMechanismクラス３０８をインスタンス化して、ExemptionMechanismインスタンスを生成する。次に、コンストラクタは、ExemptionMechanismSpiインスタンスをExemptionMechanismインスタンス内にカプセル化する。その際に、コンストラクタは、ExemptionMechanismインスタンスのメソッドをExemptionMechanismSpiインスタンスの対応するメソッドにマッピングする。本発明の一実施形態において、ExemptionMechanismインスタンスのInitメソッドがExemptionMechanismSpiインスタンスのEngineInitメソッドにマッピングされ、GenExemptionBlobメソッドがEngineGenExemptionBlobメソッドにマッピングされる。このマッピングは、ExemptionMechanismインスタンスのメソッドへの呼び出しがExemptionMechanismSpiインスタンスの正しいメソッドに送られるようにする。ExemptionMechanismSpiインスタンスがExemptionMechanismインスタンス内にカプセル化されると、ExemptionMechanismインスタンスのインスタンス化が完了する。

【００２３】その後、GetImplメソッドはCipherクラス３０６のGetInstanceメソッドに返り（これは、GetImplメソッドの第１の呼び出しからの復帰である）、GetInstanceメソッドに、実装のインスタンス、制約のセット、および（もしあれば）ExemptionMechanismインスタンスを提供する。次に、Cipherクラス３０６のGetInstanceメソッドは、Cipherクラス３０６のコンストラクタを呼び出し、コンストラクタにGetImplメソッドから受け取った実装のインスタンス、制約のセット、および（もしあれば）ExemptionMechanismインスタンスを渡す。これに対応して、コンストラクタは、Cipherクラス

３０６をインスタンス化（４４８）して、Cipherインスタンスを生成する。次に、コンストラクタは、実装のインスタンス、制約のセット、および（もしあれば）ExemptionMechanismインスタンスをCipherインスタンス内にカプセル化する（４５２）。つまり、Cipherインスタンスは「ラッパー」オブジェクトとして動作する。実装のインスタンスをCipherインスタンスにカプセル化する場合に、コンストラクタはCipherインスタンスのメソッドを対応する実装のインスタンスメソッドにマッピングする。本発明の一実施形態において、CipherインスタンスのInitメソッドは、実装のインスタンスのEngineInitメソッドにマッピングされ、Updateメソッドは、EngineUpdateメソッドにマッピングされ、DoFinalメソッドは、EngineDoFinalメソッドにマッピングされる。このマッピングは、Cipherインスタンスのメソッドへの呼び出しが正しい実装のインスタンスのメソッドに送られるようにする。これらのメソッドの実装が実装のインスタンスによって提供されるので、このようになる。カプセル化処理が完了すると、コンストラクタは、Cipherクラス３０６のGetInstanceメソッドに戻る。次に、GetInstanceメソッドは呼び出しているアプリケーション１０４に戻り、アプリケーション１０４に新しく構築されたCipherインスタンスを提供する（４５６）。その後、呼び出しているアプリケーション１０４は、Cipherインスタンスのメソッドを直接呼び出すことができる。

【００２４】本発明の一実施形態において、呼び出しているアプリケーション１０４がCipherインスタンスを呼び出す必要のある最初のメソッドの１つが、Initメソッドである。この方法は、Cipherインスタンスの初期化を行い、通常の動作用にInitメソッドを準備する。このメソッドを呼び出しているときに、呼び出しているアプリケーション１０４は、初期化パラメータのセットを提供する。本発明の一実施形態において、これらのパラメータには、暗号化に使用される暗号鍵と、暗号の繰り返し数などのアルゴリズム固有の属性を規定している任意の他の暗号パラメータ（特定の暗号化アルゴリズムが必要とする場合）が含まれる。

【００２５】Initメソッドが呼び出された場合に、Initメソッドは、呼び出しているアプリケーション１０４によって渡された初期化パラメータとCipherインスタンス内にカプセル化された制約を比較する。初期化パラメータが制約のレベルまたはそれ以下の場合、Initメソッドは初期化パラメータを実装のインスタンスのEngineInitメソッドに渡して、実装のインスタンスが初期化できるようにする。実装のインスタンスが初期化された後、Cipherインスタンスが動作可能な状態になる。このように、暗号化/復号化の動作を実行するために、CipherインスタンスのUpdateメソッドとDoFinalメソッドを、呼び出しているアプリケーション１０４よって呼び出すことができる。しかし、Initメソッドが、呼び出しているアプリケーション１０４によって渡される初期化パラメータがカプセル化された制約のレベルを超えたと判断した場合には、Initメソッドが初期化パラメータを実装のインスタンスのEngineInitメソッドに渡されないようにする。それによって、実装のインスタンス、すなわち、Cipherインスタンスを初期化しないようにする。Cipherインスタンスが初期化されない場合、Cipherインスタンスは正常に動作できなくなる。このように、初期化を行わないことによって、Initメソッドは、効果的にCipherインスタンスを動作できないようにする。この方法で、Initメソッドは、カプセル化された制約が実装のインスタンスに課せられることを確実にする施行ロジックとして働く。

【００２６】ExemptionMechanismインスタンスがCipherインスタンス内にカプセル化される場合に、Cipherクラス３０６のInitメソッドは、追加の機能を実行する。その機能は、ExemptionMechanismインスタンスがアプリケーション１０４によって正しく呼び出され、データの暗号化を行う前に必要な動作を実行することを確実にする。たとえば、免除機構がキー・リカバリーの場合には、データの暗号化を行う前に、ExemptionMechanismインスタンスを呼び出して、キー・リカバリーブロックを生成し、保存する必要がある。必要な動作がExemptionMechanismインスタンスによって実行されたことを確実にするために、Initメソッドは、ExemptionMechanismインスタンスのIsCryptoAllowedメソッドを呼び出す。本発明の一実施形態において、ExemptionMechanismインスタンスは、そのGenExemptionBlobメソッドが呼び出されたかどうかについて情報をその中に保持している（ExemptionMechanismインスタンスは、必要な免除機構の動作が実行される元になっている）。この情報は、IsCryptoAllowedメソッドを呼び出してアクセスできる。このIsCryptoAllowedメソッドが、必要な動作が実行された（すなわち、GenExemptionBlobメソッドが呼び出された）ことを示している場合、Initメソッドは実装のインスタンス、すなわち、Cipherインスタンスが初期化できるようにする。必要な動作が実行されていない場合は、Initメソッドが初期化ができないようにするので、Cipherインスタンスは動作できないようになる。従って、Initメソッドは、実装のインスタンスに制約を加えるだけではなく、免除機構が適用されることも確実にする。

【００２７】前述したように、それは、Cipherインスタンスによって提供されるサービスに加えられる制約を判断するJCESecurityManagerクラス３１６のGetCryptoPermissionメソッドである。これらの制約は、指定された制限１０８と、もしあれば、呼び出しているアプリケーション１０４に認められた許可１１０に基づいて決定される。GetCryptoPermissionメソッドの一実施形態を次に説明するが、実施形態を詳細に説明する前に、本発明の全てを円滑に理解するために、制限１０８と許可１１

0について簡単に説明する。

【0028】本発明の一実施形態において、制限108はデフォルト設定と免除設定の2セットの制限から構成される。基本的にデフォルト設定は、免除機構が実装されていない場合に暗号化アルゴリズムに加える必要のある初期制限を指定する。そして、免除設定には、免除機構が実装されている場合に暗号化アルゴリズムに加える必要のある制限を指定する。一般的に、免除機構が実装されている場合には、強固な暗号パラメータが使用できる。本発明の一実施形態において、両設定の制限は、適用される法律と規制に基づく。

【0029】制限の各設定（デフォルト設定または免除設定）は、0または1個以上のエントリから構成される。各エントリには、特定の暗号化アルゴリズムと、そのアルゴリズムに加えられる幾つかの制限を指定する。制限に関する各設定のエントリは同じ構造をしている。本発明の一実施形態において、各エントリは、次の情報を保存するフィールドまたは情報コンテナから構成される。

（1）暗号化アルゴリズム名または識別子
（2）免除機構名または識別子
（3）最大鍵長、および
（4）実行される暗号の最大繰り返し数などのその他のアルゴリズム固有の暗号の制限

【0030】本発明の目的のために、エントリはどのような望ましい形式も可能である。たとえば、各エントリを、その中にカプセル化された必要な情報を持っているオブジェクトとして実装できるし、各エントリをファイル内のテキストの組み合わせとすることもできる。正しい情報が提供されている限り、どのような望ましいの形式も使用できる。

【0031】図7に制限のデフォルト設定と免除設定の例を示す。デフォルト設定のエントリでは、免除機構のどれも指定せず、免除設定のエントリでは必ず免除機構を指定することに注意する。デフォルト設定は免除機構が実装されていない場合に加えられる制約を指定し、免除設定は免除機構が実装されている場合に加えられる制約を指定するので、このようになる。

【0032】制限のデフォルト設定の解釈は簡単である。基本的に、各エントリは、特定の暗号化アルゴリズムについての最大暗号パラメータを表している。従って、図7のようにBlowfishアルゴリズムでは、128ビットの最大鍵長が使用される。同様に、RC5アルゴリズムでは、64ビットの最大鍵長、10回の暗号の最大繰り返し数が使用される。免除設定の解釈もほぼ同じように簡単である。基本的に、免除設定の第1エントリは、キー・リカバリー免除機構がBlowfishアルゴリズムと共に実装されている場合に、最大鍵長を増加させて256ビットにできることを示している。同様に、第2エントリはキー・エスクロー免除機構がBlowfishアルゴリズムと共に実装されている場合に、最大鍵長を増加させて256ビットにできることを示している。免除設定では、同じアルゴリズム名（この場合はBlowfish）を複数のエントリに記述できることに注意されたい。これらのエントリに指定する免除機構が異なれば、同じアルゴリズム名を記述できる。

【0033】指定される制限108は、Cipherインスタンスに加えられる制約の決定で考慮される因子の一部に過ぎない。もう1つの因子は、呼び出しているアプリケーション104に認められた許可110がもし存在するとすれば、それらである。以前に説明したように、たとえば医療機関と金融機関用のアプリケーションのような種類のアプリケーションは、他のアプリケーションに比べて強固な暗号法を使用できる。医療機関と金融機関用のアプリケーションやその他のアプリケーションで、強固な暗号法を使用する権限は、アプリケーションに認められた許可110に反映される。本発明の一実施形態において、許可110は複数の形式のうちの1つを取る。1番目の形式はCryptoAllPermission情報である。アプリケーションがCryptoAllPermissionを与えられている場合、その意味はアプリケーションには可能な全ての許可が与えられていることになる。言い換えると、アプリケーションは制限されない。これは、認められることのできる最高の許可を可能にし、従って、極く少数のアプリケーションに認められる。

【0034】アプリケーションに認められるこれよりも低い許可は、特定の暗号アルゴリズムの暗号強度を強化し、または無制限に実装するための許可である。本発明の一実施形態において、この種の許可は、特定のアルゴリズム名（たとえば、Blowfish）と任意の最大パラメータの組み合わせ（たとえば、最大鍵長）を指定する。最大パラメータの組み合わせが指定された場合、暗号化アルゴリズムは指定される最大パラメータのレベルで実装され得る。最大パラメータの組み合わせが指定されなかった場合、暗号化アルゴリズムは任意のレベルで実装され得る（すなわち、アルゴリズムが制約されない）。このように、許可で128ビットの最大鍵長と共にBlowfishが指定されている場合、アプリケーションは128ビットの最大鍵長でBlowfish暗号化アルゴリズムを使用できる。許可でBlowfishが指定されているだけの場合、アプリケーションは鍵の長さに制限されずにBlowfish暗号化アルゴリズムを使用できる。これまで、最大パラメータについては、最大鍵長だけを説明した。最大パラメータは、暗号の最大繰り返し数などの別のパラメータを含むことができることに注意すべきである。このような別のパラメータは、RC5などの暗号化アルゴリズムが要求することもでき、したがって、最大パラメータの中に含めることもできる。

【0035】更に、アプリケーションに認められる他の許可は、特定の暗号化アルゴリズム（たとえば、Blowfi

shを使用したキー・リカバリー）に特定の免除機構を実装するための許可である。前述したように、免除機構を実装すると、通常、アプリケーションが強固な暗号パラメータ（たとえば、長い鍵長）を使用できるようになる。このように、免除機構を実装する許可は、非常に暗号強度を向上させる。許可が実際に制限１０８の内容に依存するかしないか、そして、免除機構の実装が使用可能かどうかが次に説明される。この点で、アプリケーションは、複数の許可を認められることも可能であることに注意しなければならない。たとえば、アプリケーションが、複数種類の免除機構を実装することもできる。その場合とその他の場合に、１つのアプリケーションが、複数の許可を認められることができる。

【００３６】次に、このような背景情報を元に、図８の流れ図によって、JCESecurityManagerクラス３１６のGetCryptoPermissionメソッドの動作について説明する。GetCryptoPermissionメソッドは、それが呼び出された場合、呼び出しているアプリケーション１０４によって要求されている暗号化アルゴリズム名（たとえば、Blowfish）を含むパラメータのセットを受け取る。呼び出しに対応して、GetCryptoPermissionメソッドは、まず、呼び出しているアプリケーション１０４を決定する（６０４）。すなわち、GetCryptoPermissionメソッドは、GetCryptoPermissionメソッドが呼び出される原因になったGetInstanceメソッドを呼び出したアプリケーション１０４を決定する。本発明の一実施形態において、GetCryptoPermissionメソッドは、この決定をコール・スタックを詳細に検討することによって行う。これは、呼び出し順序をトレースして、GetCryptoPermissionメソッドからGetImplメソッドに戻り、次にGetInstanceメソッドに戻り、次にGetInstanceメソッドを最初に呼び出しているアプリケーション１０４に戻ることにより行う。この方法のように、コール・スタックを詳細に検討することによって、GetCryptoPermissionメソッドが最初の呼び出しアプリケーション１０４を決定できる。

【００３７】呼び出しているアプリケーション１０４が決定されると、呼び出しているアプリケーション１０４が、それに認められた何らかの有効な許可を持っているかを決定する（６０８）。本発明の一実施形態において、これは何らかの許可がそもそもアプリケーション１０４に与えられたかどうかまず決定することで行われる。本発明の一実施形態において、この決定は、アプリケーション１０４に関係するファイルを確認し、この中に何らかの許可が含まれているかどうか確認することによって行われる。Ｊａｖａプログラミング環境では、アプリケーションのファイルはＪＡＲファイルに含まれ、この環境では、このＪＡＲファイルを許可の確認に使用する。

【００３８】何らかの許可が見つかった場合、検証処理が実行され、許可が有効であることが保証される。本発明の一実施形態において、この検証はデジタル署名を使用して行われる。具体的には、１または複数の許可を含む任意のアプリケーション１０４には、デジタル署名されたＪＡＲファイルが存在する。このデジタル署名は、アプリケーション１０４の出所が信頼されており、アプリケーション１０４の内容が変更されていなかったことを確実にしている。このデジタル署名が検証される場合、ＪＡＲファイル内に含まれる許可が有効であることを意味している。このデジタル署名が検証されない場合、許可は無効である。GetCryptoPermissionメソッドは、デジタル署名検証機構を使用してこの検証を行う。本発明の目的のために、適切で効果的なデジタル署名検証機構も使用できる。

【００３９】GetCryptoPermissionメソッドが呼び出しているアプリケーション１０４が有効な許可を持っていないと判断した場合に、GetCryptoPermissionメソッドは、制限のデフォルト設定の制限に基づいてCipherインスタンスに加えられる制約を決定する（６１２）。具体的には、GetCryptoPermissionメソッドは、呼び出しているアプリケーション１０４によって要求されている暗号化アルゴリズムと同じアルゴリズム名のエントリの、デフォルト設定エントリを検索する。そのエントリが見つかった後に、制約がそのエントリ内に指定される制限（たとえば、最大鍵長と他の制限）から導かれる。たとえば、呼び出しているアプリケーション１０４がBlowfishアルゴリズムの実装を要求している場合に、図７の例のように、制約は最大鍵長が１２８ビットのBlowfishとなるだろう。制約が決まった後、制約がGetCryptoPermissionメソッドによってJCESecurityクラス３１４のGetImplメソッドに返される（６１６）。

【００４０】工程６０８に戻って、GetCryptoPermissionメソッドが呼び出しているアプリケーション１０４が１または複数の有効な許可を持っていると判断した場合に、GetCryptoPermissionメソッドは、これらの許可のいずれかがCryptoAllPermissionかどうかを決定する（６２０）。CryptoAllPermissionの場合、アプリケーション１０４は制限されない。その場合、GetCryptoPermissionメソッドは、制約無しの指示をGetImplメソッドに返す（６２４）。しかし、許可がいずれもCryptoAllPermissionではない場合、GetCryptoPermissionメソッドは工程６２８に進む。

【００４１】工程６２８まで処理すると、アプリケーション１０４が１または複数の有効な許可を持っており、これらの許可がいずれもCryptoAllPermissionでないことが分かる。従って、許可は次の２種類のうちのいずれかになることを意味する。

（１）加えられるべき免除機構を要求しない種類（すなわち、特定の暗号化アルゴリズムと随意な最大パラメータのセットを指定する種類）、または、（２）加えられるべき免除機構を要求する種類（すなわち、特定の暗号

化アルゴリズムと共に免除機構を指定する種類）

【００４２】工程６２８で、GetCryptoPermissionメソッドは、許可のいずれかが、加えられるべき免除機構を要求しない種類のものであるかどうか決定する。許可のいずれかがこの種類の場合、それらの許可の各々に対して、その許可が適用できるかどうかについて決定する（６３２）。許可が適用できるのは、許可に指定された暗号化アルゴリズムが、アプリケーション１０４によって要求されている暗号化アルゴリズムと同じ場合である。たとえば、アプリケーション１０４がBlowfishアルゴリズムの実装を要求している場合、許可に指定された暗号化アルゴリズムがBlowfishアルゴリズムを指定しているときに１つの許可が適用になる。本発明の一実施形態においては、最大１つの許可が適用される。GetCryptoPermissionメソッドが許可の１つが適用になることを決定する場合、GetCryptoPermissionメソッドは、許可に指定された最大パラメータ（もし存在すれば）に基づいてCipherインスタンスに加えられる制約を決定する。すなわち、最大パラメータのセットが許可に指定された場合、制約は指定された最大パラメータに基づいて決定される。最大パラメータの組み合わせが指定されていない場合、制約は無制限になり、暗号化アルゴリズムは制約されない。制約が決定された後、制約がGetCryptoPermissionメソッドによってJCESecurityクラス３１４のGetImplメソッドに返される（６３６）。

【００４３】工程６３２に戻って、GetCryptoPermissionメソッドが、加えられるべき免除機構を要求しない許可のいずれもが適用できないことを決定した場合、工程６４０に進む。工程６４０で、GetCryptoPermissionメソッドは、アプリケーション１０４に与えられた許可のいずれかが、加えられるべき免除機構を要求する種類であるかどうか決定する。このような許可が見つからなかった場合、GetCryptoPermissionメソッドは制約のデフォルト設定を使用してCipherインスタンスに加える制約を決定する（６４４）。制約を決定する方法は、前述の工程６１２に関連して説明した方法と同じである。制約が決まった後、制約がGetCryptoPermissionメソッドによってGetImplメソッドに返される（６４８）。

【００４４】一方、アプリケーション１０４に認められた許可の少なくとも１つが、加えられるべき免除機構を要求する種類であると、GetCryptoPermissionメソッドが決定した場合には、工程６５２に進む。工程６５２で、GetCryptoPermissionメソッドは、加えられるべき免除機構を要求する許可のいずれかが適用できるかどうかを決定する。具体的には、GetCryptoPermissionメソッドは、これらの許可のどれが要求されている暗号化アルゴリズムに適用できるか、適用する許可の各々に対して、その特定の暗号化アルゴリズム/免除機構のセットが使用できるか、および指定された免除機構の実装が使用可能かどうかを、決定する。これらの機能を実行する際に、GetCryptoPermissionメソッドは、制限の免除設定を参照する。これらの動作は、例示によってよく理解できる。

【００４５】要求されている暗号化アルゴリズムがBlowfishアルゴリズムであり、アプリケーションが、次の２種類の許可を認められているとする。

（１）Blowfishとキー・ウィークニングおよび（２）Blowfishとキー・リカバリー

【００４６】さらに図７に示されている制限の免除設定を仮定する。この例では、許可が共にBlowfishに関連しているので、両方の許可が、要求しているアプリケーションに適用する。従って、両方の許可は共に処理されることになり、第１の許可から処理し始める。第１の許可は、キー・ウィークニングがBlowfishと共に使用されることができるようにする。この許可が使用可能かどうかを決定するために、GetCryptoPermissionメソッドがこの組み合わせを持つエントリの免除設定を検索する。免除設定にはBlowfishの２つのエントリが存在するが、これらのエントリのどちらも免除機構としてキー・ウィークニングを指定していない。従って、制限の免除設定が使用できるBlowfishとキー・ウィークニングの組み合わせが明示されていないないので、この許可を使用または適用できない。

【００４７】この場合、GetCryptoPermissionメソッドは、Blowfishと共に使用されるキー・リカバリーを許可する次の許可の処理に進む。この許可は最初と同じ方法すなわち、免除設定のエントリを検索して処理される。今回は、Blowfishとキー・リカバリーが指定された組み合わせを使用できるエントリが見つかる。結果として、この許可が使用又は適用可能になる。しかし、問い合わせはそこで終わらない。GetCryptoPermissionメソッドは、この許可の使用を認める前に、指定された免除機構（この例ではキー・リカバリー）の有効な実装が使用可能かどうか決定する。そして、実装が使用可能でない場合、この許可を適用しない。この決定を行う際に、GetCryptoPermissionメソッドは指定された免除機構を実装する有効な一般的な実装１０６（図４）を検索する。この処理の終了（６５２）までに、GetCryptoPermissionメソッドは認められた許可のいずれかを適用できるかどうかが分かることになる。

【００４８】許可が適用できることをGetCryptoPermissionメソッドが決定する場合に、GetCryptoPermissionメソッドは、制限のデフォルト設定ではなく免除設定を使用し、Cipherインスタンスに加えられる制約を決定する（６５６）。具体的には、GetCryptoPermissionメソッドは、この許可と同じアルゴリズム名と免除機構を持っている免除設定のエントリから制約を導く。対象の例でこのエントリは、免除設定の最初のエントリであり、その制約は最大鍵長が２５６ビットのキー・リカバリーを備えたBlowfishである。これらの制約が決まった後、制

約がGetCryptoPermissionメソッドによってJCESecurityクラス３１４のGetImplメソッドに返される（６６０）。前述したように、免除設定のエントリは、通常、デフォルト設定よりも強固な暗号パラメータが使用できるようにする。従って、免除設定の制約を導くことで、GetCryptoPermissionメソッドは、Cipherインスタンスの暗号強度を向上させる。

【００４９】工程６５２に戻って、いずれの許可も適用できない場合、GetCryptoPermissionメソッドは制限のデフォルト設定を使用して、Cipherインスタンスに加えられる制約を決定する（６４４）。制約を決定する方法は、工程６１２に関連して上記で説明した方法と同じである。従って、アプリケーション１０４は、アプリケーションがまったく許可を与えられていない場合と同じように取り扱われる。制約が決まった後に、制約がGetCryptoPermissionメソッドによってGetImplメソッドに返される（６４８）。説明した方法のように、GetCryptoPermissionメソッドはCipherインスタンスに加えられる制約を決定する。最初に許可を適用しようとし、次に、いずれの許可も適用しない場合に初期制限を使用することによって、GetCryptoPermissionメソッドは、Cipherインスタンスに、与えられる限りの制限で最大の暗号強度を与えようとする。言い換えると、GetCryptoPermissionメソッドは最低レベルの制約を加えようとする。

【００５０】前述したように、制限１０８の全セット（図１）を含む制限のデフォルト設定と免除設定は、適用可能な法律と規制に基づいている。本発明の一実施形態において、それらは少なくとも次の２つの法律と規制に基づいて導かれる。

（１）米国輸出法、および（２）現地法（フレームワーク１０２が輸入される国または地域の法律）

【００５１】これらの法律のセットは、ほとんどの場合、異なるので、両方の法律のセットと一致する１つの制限のセットを導くために、調整処理が実行される。本発明の一実施形態において、この調整はマージ処理を使用して行われる。具体的には、２つの法律のセットはマージされて制限１０８の結果セットを生成し、マージは、その得られた制限１０８が２つの法律のセットの最も制約された制限を含むような方法で実行される。最も制約された制限を選択することで、マージ処理は得られた制限１０８が両方の法律のセットに従っていることを保証する。

【００５２】図９はマージ処理の概要を表している。図に示されているように、米国輸出法７０２は、デフォルトコンポーネント７０６と免除コンポーネント７０８から構成される。同様に、現地法７０４は、デフォルトコンポーネント７１０と免除コンポーネント７１２から構成される。デフォルトコンポーネント７０６と７１０は、免除機構が実装されていない場合に暗号化アルゴリズムに適用されるデフォルトの制限を指定する。そして、免除コンポーネント７０８と７１２は、免除機構が実装されている場合の制限を指定する。本発明の一実施形態において、デフォルトコンポーネント７０６、７１０と免除コンポーネント７０８、７１２は前に、図7と関連して説明した制限のデフォルト設定７１４と免除設定７１６と同じ形式をしている。すなわち、各コンポーネント７０６、７１０、７０８、７１２は、０またはそれ以上のエントリから構成される。各エントリは、以下を保存するためのフィールドまたはコンテナを備える。

（１）暗号化アルゴリズム名または識別子

（２）免除機構名または識別子

（３）最大鍵長および

（４）他の暗号制限

【００５３】得られた制限１０８を導くには、デフォルトコンポーネント７０６と７１０をエントリごとにマージして、得られる制限１０８のデフォルト設定７１４を生成する。また、免除コンポーネント７０８と７１２をエントリごとにマージして、得られる制限１０８の免除設定７１６を生成する。この制限が導かれた後、得られた制限１０８はJCESecurityManagerクラス３１６のGetCryptoPermissionメソッドによって使用されて、Cipherインスタンスに加えられる制約を決定する。

【００５４】次に、図１０及び図１１のフローチャートに沿って、マージ処理の一実施形態を説明する。次の説明では、政策Ａ、ＢおよびＣを使用して説明する。政策ＡとＢは、マージの情報源（たとえば、米国輸出法と現地法）を指す。また、政策Ｃはマージ結果を指す（たとえば、得られた制限１０８）。図９に示したように、デフォルトコンポーネント７０６、７１０と免除コンポーネント７０８、７１２は別個のマージ操作を使用して別々にマージされる。しかし、両方のマージで同じマージ処理が使用されることに注意されたい。

【００５５】さて、図１０のように、マージ処理は政策Ａの次のエントリ（この場合は最初のエントリ）の選択（８０４）から始まる。選択したエントリと政策Ｂのエントリとを比較して、対応するエントリが政策Ｂに存在するかどうかを決定する（８０８）。本発明の一実施形態において、この決定は、選択されたエントリのアルゴリズム名および免除機構名と、政策Ｂのエントリのアルゴリズム名および免除機構名とを比較して行われる。政策Ｂのエントリに同じ名前のアルゴリズムと免除機構の組み合わせが存在すると、そのエントリが対応するエントリになる。この場合に、２つの対応するエントリの制限が比較されて、最も制約された制限を決定する（８２０）。

【００５６】この方法の例として、政策ＡとＢの両方のアルゴリズム名がＲＣ５で、免除機構が存在しないエントリを考える。また、政策Ａのエントリの最大鍵長が６４ビット、最大繰り返し数が１２、政策Ｂのエントリの最大鍵長が１２８ビット、最大繰り返し数が１０とす

る。この場合に、最も制約された制限は最大鍵長が６４ビット、最大繰り返し数が１０になる。この例に示したように制限ごとに最も制約された制限が決定される。

【００５７】最も制約された制限が決まった後、新しいエントリが政策Ｃに生成される（８２４）。この新しいエントリには、２つの対応するエントリと同じアルゴリズム名と免除機構名が存在する。また、この新しいエントリには、その制限として、工程８２０で決定された最も制約された制限が存在する。新しいエントリが政策Ｃに生成されると、現在選択しているエントリの処理が終了する。そして、政策Ａにこれ以上エントリが存在するかどうかの判断がなされる（８２８）。エントリが存在する場合には、処理を工程８０４に戻して、政策Ａの次のエントリを選択し、処理する。エントリが存在しない場合には、処理を工程８３２に進める。

【００５８】工程８０８に戻って、政策Ａで選択したエントリに対応するエントリが政策Ｂに存在しないと判断された場合、政策Ｂにワイルドカードのエントリが存在しないかどうかの判断がなされる（８１２）。このワイルドカードは、政策Ｂに明示して示されていない全てのアルゴリズム名/免除機構の組み合わせのための容器として動作する。ワイルドカードが政策Ｂに見つからなかった場合、選択されたエントリの処理が終了する。政策Ｃには、新しいエントリが生成されず、処理は、政策Ａの次のエントリを検索するために工程８２８へ進む。

【００５９】一方、政策Ｂにワイルドカードのエントリが存在すると判断された場合、選択されたエントリの制限とワイルトカードのエントリの制限を比較して、最も制約された制限を決定する（８１６）。この決定は、工程８２０について前述した説明と同じ方法で行われる。最も制約された制限が決まった後、新しいエントリが政策Ｃに生成される（８２４）。この新しいエントリには、選択したエントリと同じアルゴリズム名と免除機構名が存在する。また、この新しいエントリには、その制限として、工程８１６で決まった最も制約された制限が存在する。新しいエントリが政策Ｃに生成されると、現在選択しているエントリの処理が終了する。そして、更にエントリが政策Ａに存在するかどうかの判断がなされる（８２８）。エントリが存在する場合には、処理を工程８０４に戻して、政策Ａの次のエントリを選択し、処理する。この処理は、政策Ａの全エントリが処理されるまで続く。

【００６０】政策Ａの全エントリが処理された後、政策Ａのエントリに対応しない政策Ｂの全エントリを処理する番になる。しかし、これを行う前に、政策Ａにワイルドカードのエントリが存在するかどうかを判断する（８３２）。政策Ａがワイルドカードのエントリを持っていない場合、政策Ｂの追加のエントリは、それらが政策Ｃに作成される追加のエントリになることはないので、これ以上処理される必要はない。このようにして、政策Ａにワイルドカードのエントリが存在しない場合、政策Ｃの構築が終了する（８３６）。

【００６１】一方、政策Ａにワイルドカードのエントリが存在する場合、政策Ｂの処理は、政策Ｂの次のエントリ（この場合は最初のエントリ）の選択から始まる（８４０）。選択したエントリと政策Ｃのエントリとを比較して、対応するエントリが政策Ｃに存在するかどうかを判断する（８４４）。本発明の一実施形態において、この決定は、選択されたエントリのアルゴリズム名および免除機構名、ならびに政策Ｃのエントリのアルゴリズム名および免除機構名とを比較することによって行われる。対応するエントリが政策Ｃに見つかった場合、選択されたエントリは、政策Ａのエントリの処理の一部としてすでに処理されたことを意味している。この場合に、選択しているエントリの処理を必要としない。結果として、処理が工程８５６に進んで政策Ｂの次のエントリを検索する。

【００６２】一方、選択されたエントリが政策Ｃのいずれのエントリにも対応していない場合、選択されたエントリの制限と政策Ａのワイルトカードのエントリの制限を比較して最も制約された制限を決定する（８４８）。この決定は、工程８２０について前述した説明と同じ方法で行われる。最も制約された制限が決定された後、新しいエントリが政策Ｃに生成される（８５２）。この新しいエントリには、選択されたエントリと同じアルゴリズム名と免除機構名が存在することになる。また、この新しいエントリには、その制限として工程８４８で決定した最も制約された制限が存在することになる。新しいエントリが政策Ｃに生成された後に、現在選択されたエントリの処理が終了する。そして、政策Ｂに更にエントリが存在するかどうかを決定する（８５６）。エントリが存在する場合、処理を工程８４０に戻して政策Ｂの次のエントリを選択し、処理する。この処理は、政策Ｂの全エントリが処理されるまで続く。全エントリが処理されると、政策Ｃの構築が終了する（８６０）。

【００６３】本発明の一実施形態において、説明されたマージ処理はJCESecurityクラス３１４のイニシャライザによって実行される。JCESecurityクラス３１４が呼び出されるとすぐに、このイニシャライザは呼び出される。イニシャライザが呼び出された場合、イニシャライザは、イニシャライザに提供された法律の２つ以上のセットをマージして、制限の全セット１０８を生成する。生成されるのはこの制限の全セット１０８（デフォルト設定と免除設定を備える）であり、これは、Cipherインスタンスに加えられる制約を決定するGetCryptoPermissionメソッドによってその後に使用される。

【００６４】前述したように、JCESecurityクラス３１４のGetImplメソッドが、関連する一般的な実装１０６のインスタンス化を担当して、実装のインスタンスを生成する。インスタンス化処理の一部として、GetImplメ

ソッドは認証処理を実行する。本発明の一実施形態において、この認証処理は、GetImplメソッドが、関連する一般的な実装１０６を認証し、関連する一般的な実装１０６がフレームワーク１０２を認証するという相互認証の形式になる。本発明の一実施形態において、この相互認証が生じることを可能にするには、（１）関連する一般的な実装１０６のＪＡＲファイルがデシタル署名され、（２）フレームワーク１０２のＪＡＲファイルがデシタル署名され、（３）JCESecurityクラス３１４が、関連する一般的な実装のＪＡＲファイルの署名の検証に使用できる不明瞭な信頼公開鍵(obfuscated trusted public keys)のセットを埋め込んであり、（４）関連する一般的な実装１０６は、フレームワークのＪＡＲファイルの署名の検証に使用する信頼公開鍵のセットを埋め込んである。

【００６５】この前提を与えられて、相互認証が次のように実行される。まず、JCESecurityクラス３１４に埋め込まれた不明瞭な信頼公開鍵を使用して、GetImplメソッドが関連する一般的な実装のＪＡＲファイルのデジタル署名を検証する。このデジタル署名が検証される場合に、GetImplメソッドは、関連する一般的な実装１０６をインスタンス化し、この関連する一般的な実装のコンストラクタが呼び出されるようにする。コンストラクタが呼び出された場合、コンストラクタは、関連する一般的な実装１０６に埋め込まれた信頼公開鍵を使用して、フレームワークのＪＡＲファイルのデジタル署名を検証する。フレームワークのＪＡＲファイルのデジタル署名が正しいことをコンストラクタが決定する場合、コンストラクタは要求された実装のインスタンスを構築することになる。デジタル署名が正しくない場合、コンストラクタはエラーを返す。この説明に示されているように、関連する一般的な実装１０６とフレームワーク１０２の両方が正しい場合だけ、実装のインスタンスが構築されることになる。

【００６６】この検証処理の実行で、GetImplメソッドは外部のデジタル署名検証機構を信頼する。すなわち、本発明の一実施形態において、GetImplメソッドは署名の検証そのものは実行しない。逆に、GetImplメソッドは、関連する一般的な実装１０６のデジタル署名と不明瞭な信頼公開鍵を外部のデジタル署名検証機構に提示して検証を受ける。本発明の一実施形態において、外部のデジタル署名検証機構はJava Runtimeの署名機構(Signature Mechanism)になる。この署名機構は全Ｊａｖａ環境の一部であり、フレームワーク１０２の一部ではない。従って、フレームワーク１０２から見ると、この署名機構は「信頼された」コンポーネントではない。結果として、署名機構が正しく信頼可能な結果を提供すると信頼する前に、署名機構自体が適法である（すなわち、正しい検証機能を実行している）ことを保証するために検証される。

【００６７】それが署名機構を検証できるようにするために、JCESecurityクラス３１４には少なくとも２つのデジタル署名をその中に埋め込んである。１つは、不明瞭な信頼公開鍵を使用して検証できることが分かっており、もう１つは不明瞭な信頼公開鍵を使用して検証できないことが分かっている。これらの署名は、署名機構に予測できない順序で提示されて、その適法性を試験する。署名機構を試験する処理の可能な１つの実施形態が、図１２に示されている。

【００６８】図１２に示されているように、検証処理は、署名機構に提示するデジタル署名（検証可能または、検証不可能なデジタル署名）の決定（９０４）から始まる。この決定は、署名機構とって予測不能な方法で行われ、本発明の一実施形態においては、ランダム処理を使用して行われる。たとえば、ランダムな数が生成され、ランダムな数がある範囲（たとえば、０に一致）の場合、署名の１つが選択されることになる。ランダムな数が別の範囲（たとえば、１に一致）の場合、別の署名が選択されることになる。本発明の一実施形態において、工程９０４の決定は、以前に選択された署名を考慮しても行われる。以前に選択された署名が全て同じ署名の場合、工程９０４によって他の署名が選択される。これは、２つの署名の各々が少なくとも１つ選択され、署名機構の適法性を完全に試験することを保証する。

【００６９】署名の１つが選択された後、選択された署名と不明瞭な信頼公開鍵が検証のために署名機構に提示される（９０８）。次に、署名機構は署名が検証されたか、検証されなかったかのどちらかを示す応答を提供する。この応答が受け取られ（９１２）、正確性が確認される（９１６）。具体的には、署名機構に提示された署名が検証可能であった場合、その応答が、署名が検証されたことを示す指標により確認される。署名機構に提示された署名が検証不可能であった場合、その応答が、署名が検証されていないことを示す指標により確認される。提示された署名に対して受け取った応答が正しくない場合、署名機構が適法でないことが決まる（９２０）。この場合に、検証処理が終了する（９２４）。

【００７０】一方、提示された署名に対して受け取った応答が正しい場合、検証処理がｎ回実行されたかどうかに関しての決定を行う（９２８）。ここで、ｎは任意の望ましい数（たとえば、５）である。ｎ回実行されていない場合、処理を工程９０４に戻してもう１度署名を署名機構に提示し、応答を試験する。処理をｎ回実行した場合、処理を工程９３２に進める。工程９３２まで処理すると、署名機構が正しい応答をそれぞれずべての提示された署名に対して提供したことが分かる（応答が正しくない場合、処理が工程９３２まで至らずに、工程９２４で終了することになる）。このようにして、署名機構が適法であることが決まる（９３２）。この場合に、署名機構は、関連する一般的な実装１０６を認証するGetI

mplメソッドによって信頼されてよい。署名機構が適法であることが検証されると、検証処理が終了する（９３６）。

【００７１】上記の処理の成果は、検証可能なデジタル署名と検証不可能なデジタル署名が署名機構に予測不可能な順序で提示されることである。この提示順序を予測不可能にすることにより、検証処理は、不正な署名機構が、正しい応答の「偽造」をすることは、仮にそれが不可能でないとしても、極めて困難である。従って、この検証処理は、外部の署名機構の適法性を試験する効率的な方法を提供する。

【００７２】これまで検証処理は、デジタル署名検証機構に関連させて説明してきた。しかし、処理はデジタル署名検証機構に限定されないことに注意しなければならない。逆に、検証処理は一般的に適用されて、任意の信頼されていない機構の適法性を試験する。分かっている正しい応答について少なくとも２つの異なった情報設定が存在する限り、処理は、信頼されていない機構の適法性を試験するために適用される。検証処理が任意の信頼されていない機構に一般的に適用される方法を、図１３の流れ図に示す。

【００７３】図１３に示されているように、検証処理は少なくとも２つの情報設定のいずれかを信頼されていない機構に提示する決定から始まる（１００４）。この決定の工程１００４は信頼されていない機構にとって予測できない方法で行われ、本発明の一実施形態においては、ランダム処理を使用して行われる。たとえば、ランダムな数が生成され、ランダムな数がある範囲内（たとえば、０に一致）の場合、最初の情報設定が選択されることになる。また、ランダムな数が他の範囲内（たとえば、１に一致）の場合、もう一つの情報設定が選択されることになる。本発明の一実施形態において、決定の工程１００４は以前に選択された情報設定も考慮する。以前に選択された選択が全て同じ情報設定であった場合、工程１００４により他の情報設定が選択される。これは、情報設定の各々が少なくとも１度選択され、信頼されていない機構の適法性を完全に試験することを保証している。

【００７４】情報設定の１つが選択された後、選択された情報設定を信頼されていない機構に提示する（１００８）。次に、信頼されていない機構は、提示された情報設定に対する応答を提供する。この応答を受け取って（１０１２）、正確性の確認を行う（１０１６）。具体的には、各情報設定の正しい応答が分かるようになる。受け取った応答が提示された情報設定の正しい応答でない場合、信頼されていない機構が適法でないことが決定される（１０２０）。この場合に、検証処理が終了する（１０２４）。

【００７５】一方、受け取った応答が提示された情報設定の正しい応答の場合、検証処理がｎ回実行されたかどうかに関しての決定を行う（１０２８）。ここで、ｎは任意の望ましい数（たとえば、５）である。ｎ回実行されていない場合、処理を工程１００４に戻して、もう１度情報設定を信頼されていない機構に提示し、応答を試験する。処理がｎ回実行されている場合、処理は工程１０３２に進む。工程１０３２まで処理すると、信頼されていない機構が正しい応答をそれぞれ全ての提示された情報設定に対して提供したことが分かる（応答が正しくない場合は、処理が工程１０３２には至らずに工程１０２４で終了することになる）。このようにして、信頼されていない機構が適法であることが決定される（１０３２）。信頼されていない機構が適法であることが検証されると、検証処理が終了する（１０３６）。

【００７６】上記の処理の成果は、２つの情報設定が信頼されていない機構に予測不可能な順序で提示されることである。この提示順序を予測不可能にすることにより、検証処理は、不正な信頼されていない機構が、正しい応答の「偽造」をすることは、仮にそれが不可能でないとしても、極めて困難である。従って、この検証処理は、任意の信頼されていない機構の適法性を試験するための効率的な方法を提供する。

【００７７】[ハードウェアの概要]本発明の一実施形態において、本発明は、１または複数のプロセッサが実行可能な命令セットとして実装される。本発明は、カルフォルニア州マウンテンビューのSun Microsystems, Inc.社製のＪａｖａ（登録商標）プログラミングシステムを含む、これに限定されないオブジェクト指向プログラミングシステムの一部として実装し得る。図１４に、発明の実施形態が実装されているコンピュータシステム１１００のハードウェアのブロック図を示す。コンピュータシステム１１００は、情報の通信に使用するバス１１０２もしくは他の通信機構、ならびにバス１１０２に接続されて情報を処理するプロセッサ１１０４を含む。コンピュータシステム１１００には、ランダムアクセスメモリ（RAM）または他の動的記憶装置などの主記憶装置１１０６も組み込まれており、バス１１０２に接続されてプロセッサ１１０４が実行する情報と命令を保存する。また、主記憶装置１１０６は、プロセッサ１１０４が命令の実行中に使用する一時的変数または他の中間情報を保存する場合にも使用される。また、コンピュータシステム１１００は、バス１１０２に接続されてプロセッサ１１０４が使用する静的な情報と命令を保存する読み出し専用メモリ（ROM）１１０８または他の静的記憶装置を含む。磁気ディスクまたは光ディスクなどの記憶装置１１１０は、バス１１０２に接続されて情報と命令の保存に使用される。

【００７８】コンピュータシステム１１００は、バス１１０２を経由して陰極線管（ＣＲＴ）などのディスプレイ１１１２に接続され、コンピュータユーザーに情報を表示する。英数字キーおよび他のキーをはじめとする入

力装置１１１４はバス１１０２に接続されて、情報とコマンドの選択をプロセッサ１１０４に送る。キーとは異なるユーザー入力装置にマウス、トラックボール、またはカーソル方向キーなどのカーソル制御１１１６があり、方向情報とコマンドの選択をプロセッサ１１０４に送り、ディスプレイ１１１２のカーソルの動きを制御する。この入力装置は、通常第１軸（たとえば、ｘ）と第２軸（たとえば、ｙ）の２軸の２度の自由度を持っており、平面内の位置を指定できる。

【００７９】一実施形態に従って、本発明の機能は、主記憶装置１１０６に格納された１または複数の命令の１または複数のシーケンスを実行するプロセッサ１１０４に応じてコンピュータシステム１１００が提供する。このような命令は、記憶装置１１１０などのコンピュータが読み込み可能な主記憶装置以外の媒体から主記憶装置１１０６に読み込まれる。主記憶装置１１０６に格納された命令シーケンスを元に、プロセッサ１１０４がここで説明された処理ステップを実行する。別の実施形態として、ハードワイヤード回路を、発明を実装するソフトウェア命令に代えて使用することもできる。また、ハードワイヤード回路を、発明を実装するソフトウェア命令と組み合わせて使用することもできる。このように、発明の実施形態はハードウェア回路とソフトウェアの任意の組み合わせに限定されない。

【００８０】ここで使用されている「コンピュータが読み取り可能な媒体」という用語は、実行に使用するプロセッサ１１０４に命令を提供することに関係する任意の媒体を示す。このような媒体は、不揮発性媒体、揮発性媒体および伝送媒体に限定されないが、それらを含む多数の形式がある。たとえば、不揮発性媒体には、記憶装置１１１０などの光ディスクまたは磁気ディスクがある。揮発性媒体には、主記憶装置１１０６などの動的記憶装置がある。伝送媒体には、バス１１０２を構成する配線をはじめとする同軸ケーブル、銅線、および光ファイバーがある。また、伝送媒体は、音波または電磁波、たとえば電波、赤外線、および光データ通信中に生成される波などの形式にもなる。

【００８１】たとえば、コンピュータが読み取り可能な媒体の一般的な形式には、フロッピー（登録商標）ディスク、フレキシブルディスク、ハードディスク、磁気テープもしくはその他の磁気媒体、ＣＤ－ＲＯＭ、その他の光媒体、パンチカード、紙テープ、その他のせん孔式の物理的媒体、ＲＡＭ、ＰＲＯＭおよびＥＰＲＯＭ、ＦＬＡＳＨ－ＥＰＲＯＭ、その他のメモリーチップもしくはカートリッジ、後述する搬送波、またはその他のコンピュータが読み込める媒体がある。

【００８２】コンピュータが読み取り可能な媒体のいろいろな種類には、１または複数の命令の１または複数のシーケンスを、命令を実行するプロセッサ１１０４に運ぶことも含まれる。たとえば、命令は最初にリモートコンピュータ上の磁気ディスクに運ばれる。リモートコンピュータは、命令をその動的記憶装置にロードし、モデムを使用して電話線上に命令を送信する。コンピュータシステム１１００に対してローカルなモデムは電話線でデータを受信し、赤外線トランスミッタを使用してデータを赤外線信号に変換する。赤外線検出器は赤外線信号のデータを受信し、適当な回路がデータをバス１１０２上に配置する。バス１１０２は、データを主記憶装置１１０６に運ぶ。そして、プロセッサ１１０４が命令を取り出し実行する。主記憶装置１１０６が受け取った命令は、プロセッサ１１０４が実行する前か後のどちらかに随意で記憶装置１１１０に保存される。

【００８３】コンピュータシステム１１００は、また、バス１１０２に接続された通信インターフェイス１１１８を含む。通信インターフェイス１１１８は２方向のデータ通信が可能であり、ローカルネットワーク１１２２に接続するためのネットワークリンク１１２０とも繋がっている。たとえば、通信インターフェイス１１１８は、デジタル総合サービス網（ＩＳＤＮ）カードまたはデータ通信接続を対応する種類の電話線に提供するモデムとすることもできる。また、その他の例として、通信インターフェイス１１１８は、データ通信接続を互換ＬＡＮに提供するローカルエリアネットワーク（ＬＡＮ）カードとすることもできる。無線リンクも実装できる。このような実装で、通信インターフェイス１１１８は、いろいろな種類の情報であるデジタルデータの流れを運搬する電気、電磁気または光信号を送受信する。

【００８４】ネットワークリンク１１２０は、通常、１または複数のネットワークを使用して他のデータ装置に対するデータ通信を可能とする。たとえば、ネットワークリンク１１２０は、ローカルネットワーク１１２２を使用してホストコンピュータ１１２４またはインターネットサービスプロバイダ（ＩＳＰ）１１２６によって運営されているデータ装置に対して接続を提供する。次に、ＩＳＰ１１２６は一般に「インターネット」１１２８と呼ばれている世界的なパケットデータ通信ネットワークを使用してデータ通信サービスを提供する。ローカルネットワーク１１２２とインターネット１１２８は、共にデジタルデータの流れを運搬する電気信号、電磁気信号または光信号を使用する。いろいろなネットワークを使用する信号ならびにネットワークリンク１１２０上の信号および通信インターフェイス１１１８を使用する信号は、情報を運ぶ典型的な搬送波の形をしている。これらの信号は、デジタルデータを、コンピュータシステム１１００に運搬し、コンピュータシステム１１００から運搬する。

【００８５】コンピュータシステム１１００は、ネットワーク、ネットワークリンク１１２０および通信インターフェイス１１１８を使用してメッセージを送信でき、プログラムコードをはじめとするデータを受信できる。

インターネットでは、アプリケーションプログラムのコードが要求されたとき、サーバー１１３０がそれをインターネット１１２８、ＩＳＰ１１２６、ローカルネットワーク１１２２および通信インターフェイス１１１８を使用して送信できる場合がある。受信したコードは、コードを受け取った際にプロセッサ１１０４で実行したり、或いは、記憶装置１１１０または他の不揮発性記憶装置に保存して後で実行したりできる。この方法で、コンピュータシステム１１００は搬送波の形式でアプリケーションコードを取得できる。

【００８６】現時点で、発明は特別の実施形態に基づいて説明されているが、それに限定されない。発明の精神から逸脱することなく、この開示の利益を使用して当業者により種々の変更が行える。従って、本発明は、本発明を記述するために使用されている特定の実施形態に制限されるものではなく、特許の請求範囲に基づいてのみ限定される。

【００８７】

【発明の効果】以上の如く本発明によれば、要求された実装の動的な構築により、フレームワークは、必要な制約がアプリケーションに提供されるサービスに課されることを確実なものとする。

【図面の簡単な説明】

【図１】本発明の一実施形態に係るシステム全体を示すブロック図である。

【図２】図１のシステム全体の一般的な動作を示す流れ図である。

【図３】本発明の一実施形態を示す詳細なブロック図である。

【図４】本発明の一実施形態を示す詳細なブロック図である。

【図５】図３及び図４の実施形態の動作を示す流れ図である。

【図６】図３及び図４の実施形態の動作を示す流れ図である。

【図７】デフォルト設定と免除設定をはじめとする制限の組み合わせの例を表した図である。

【図８】JCESecurityManagerオブジェクトクラスのGetCryptoPermissionメソッドの一実施形態の動作を示す流れ図である。

【図９】規則の複数のセットを制限の１つのセットにマージする処理の概要を示す本発明の一実施形態の流れ図である。

【図１０】規則の複数のセットを制限の一つのセットにマージする方法を示す本発明の一実施形態の流れ図である。

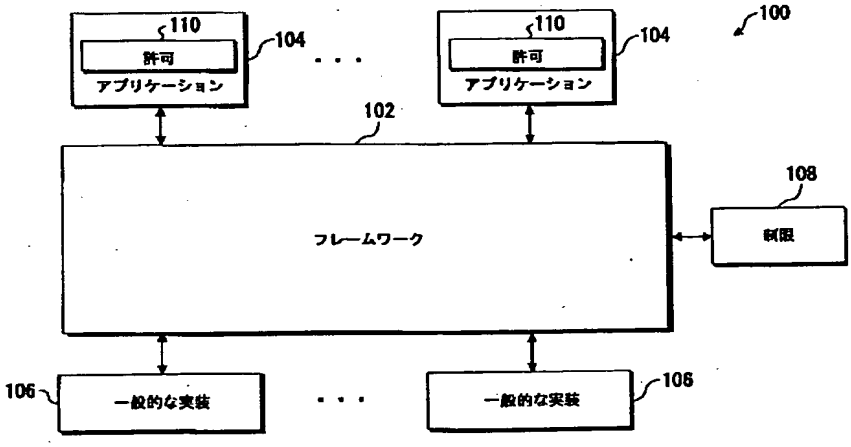
【図１１】規則の複数のセットを制限の一つのセットにマージする方法を示す本発明の一実施形態の流れ図である。

【図１２】信頼されていないデジタル署名検証機構の適法性を試験する方法を示す本発明の一実施形態の流れ図である。

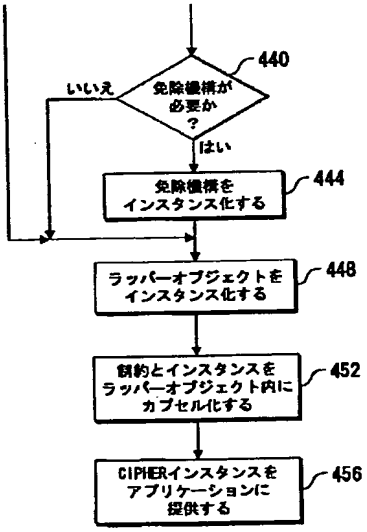
【図１３】任意の信頼されていない機構の適法性を試験する方法を示す本発明の一実施形態の流れ図である。

【図１４】本発明が実装するコンピュータシステムのハードウェアブロック図である。

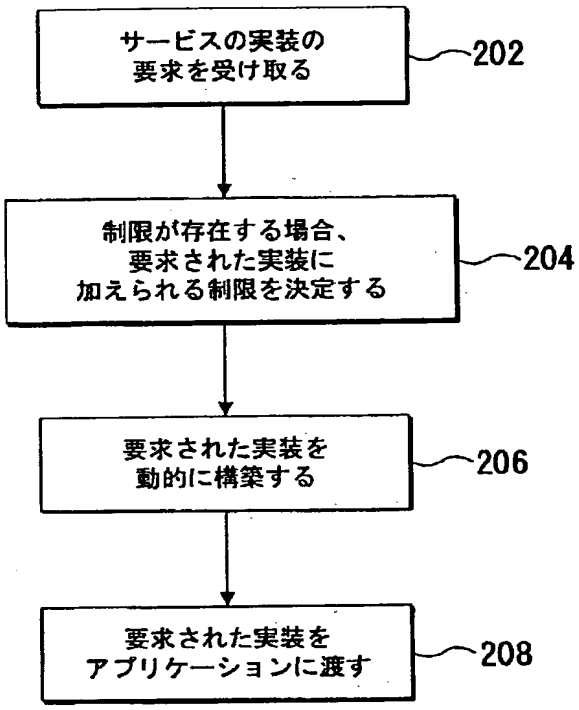
【図1】



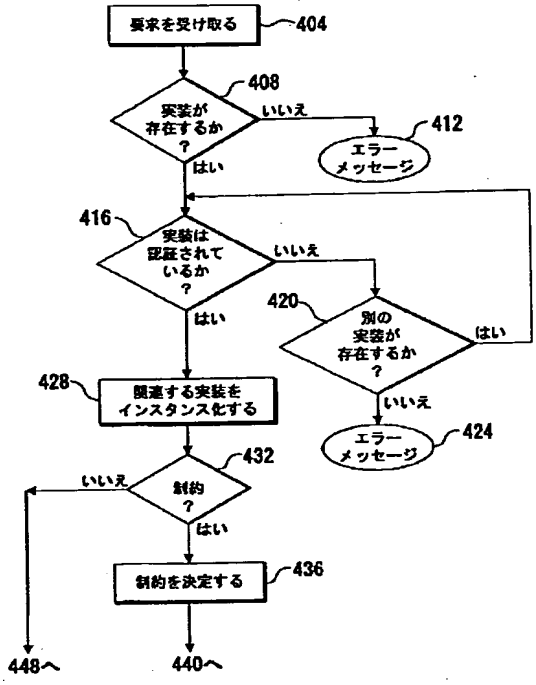
【図6】



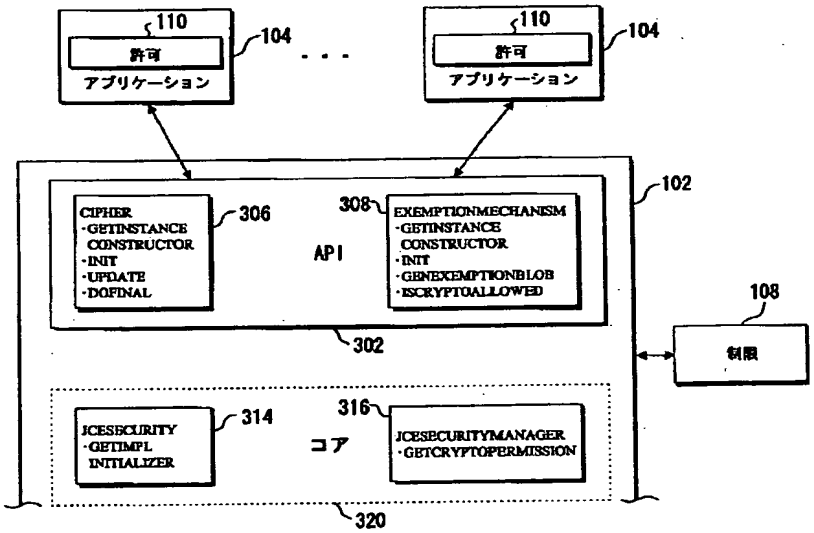
【図2】



【図5】



【図3】



【図7】

デフォルト設定

(1) アルゴリズム名:BLOWFISH
免除機構:
最大鍵長:128ビット
他の制限:

(2) アルゴリズム名:DES
免除機構:
最大鍵長:128ビット
他の制限:

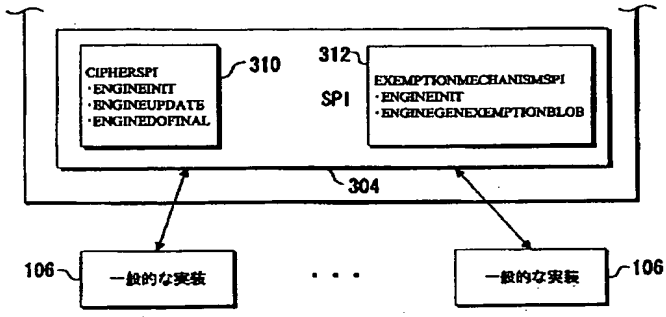
(3) アルゴリズム名:RC5
免除機構:
最大鍵長:64ビット
他の制限:10回

免除設定

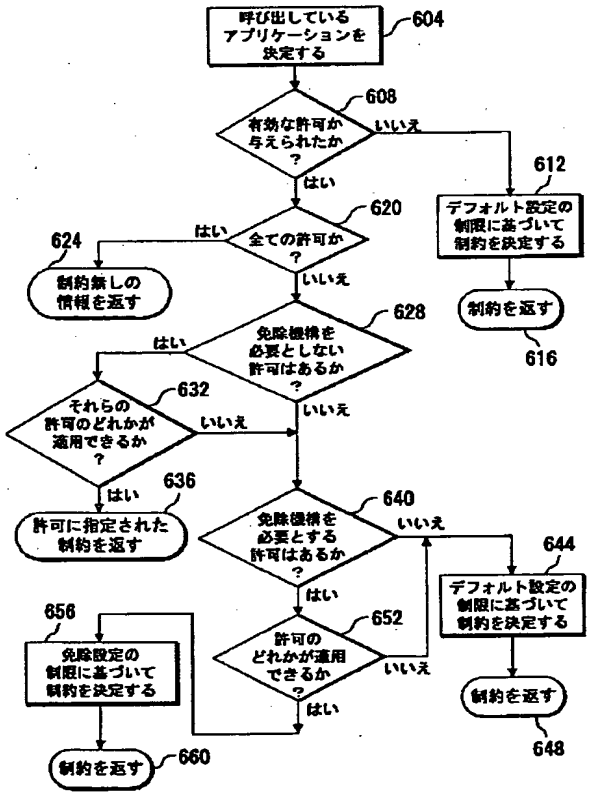
(1) アルゴリズム名:BLOWFISH
免除機構:キー・リカバリ
最大鍵長:256ビット
他の制限:

(2) アルゴリズム名:BLOWFISH
免除機構:キー・エスクロー
最大鍵長:256ビット
他の制限:

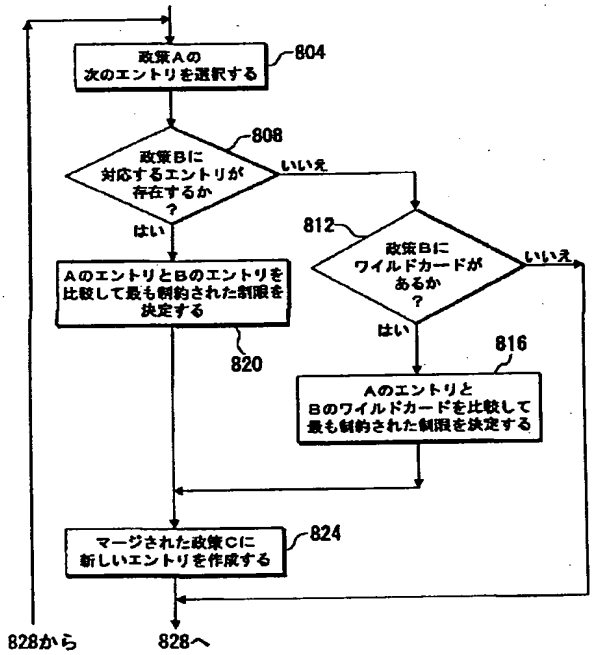
【図4】


CIPHERSPI
・ENGINEINIT
・ENGINEUPDATE
・ENGINEDOFINAL
310
312
SPI
EXEMPTIONMECHANISMSPI
・ENGINEINIT
・ENGINEGENEXEMPTIONBLOB
304
106
一般的な実装
・・・
一般的な実装
106

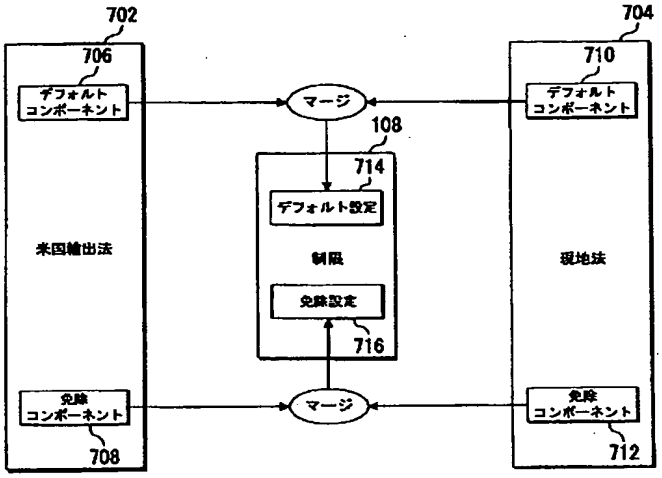
【図8】


呼び出している
アプリケーションを
決定する
604
有効な許可か
与えられたか
?
608
いいえ
はい
デフォルト設定の
制限に基づいて
制約を決定する
612
制約を返す
616
全ての許可か
?
620
はい
624
制約無しの
情報を返す
いいえ
免除機構を
必要としない
許可はあるか
?
628
はい
それらの
許可のどれかが
適用できるか
?
632
いいえ
はい
許可に指定された
制約を返す
636
免除機構を
必要とする
許可はあるか
?
640
いいえ
はい
デフォルト設定の
制限に基づいて
制約を決定する
644
制約を返す
648
許可の
どれかが適用
できるか
?
652
いいえ
はい
免除設定の
制限に基づいて
制約を決定する
656
制約を返す
660

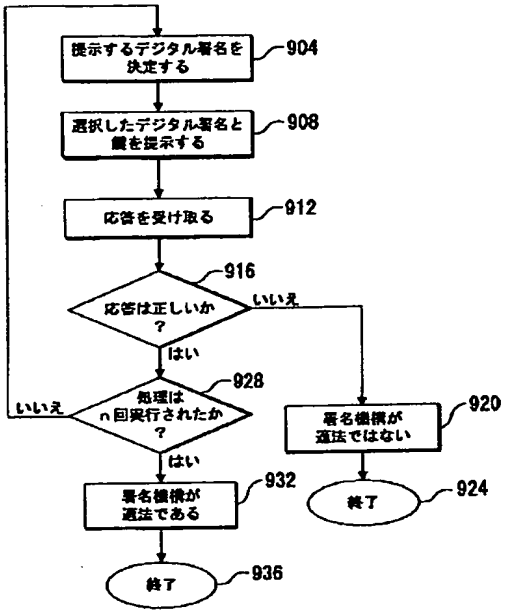
【図10】


政策Aの
次のエントリを選択する
804
政策Bに
対応するエントリが
存在するか
?
808
いいえ
はい
AのエントリとBのエントリを
比較して最も制約された制限を
決定する
820
政策Bに
ワイルドカードが
あるか
?
812
いいえ
はい
816
Aのエントリと
Bのワイルドカードを比較して
最も制約された制限を決定する
マージされた政策Cに
新しいエントリを作成する
824
828から
828へ

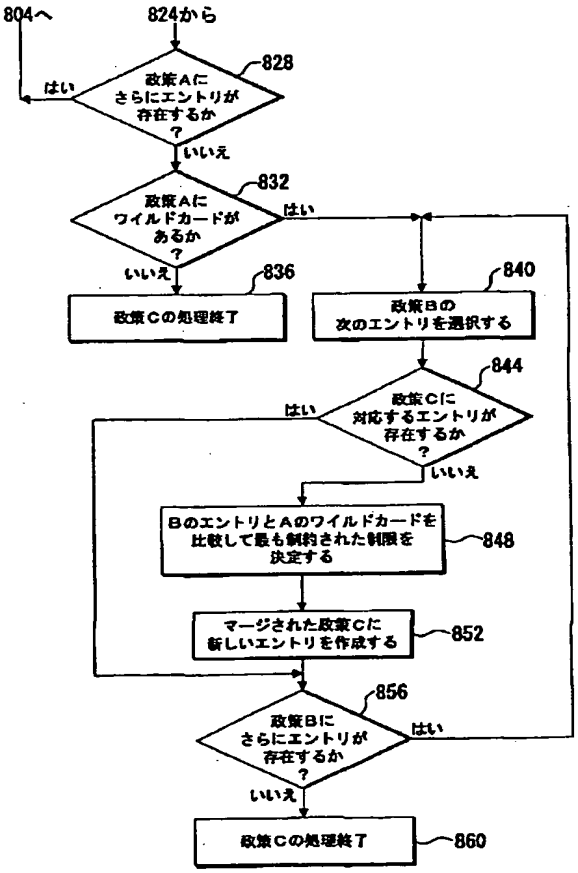
【図9】



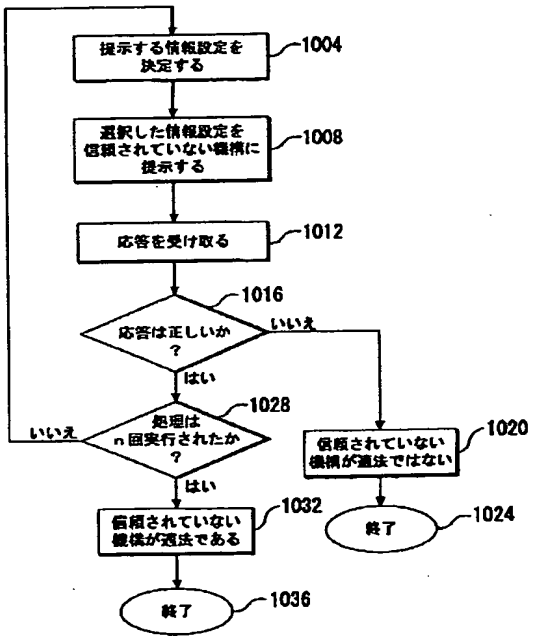
【図12】



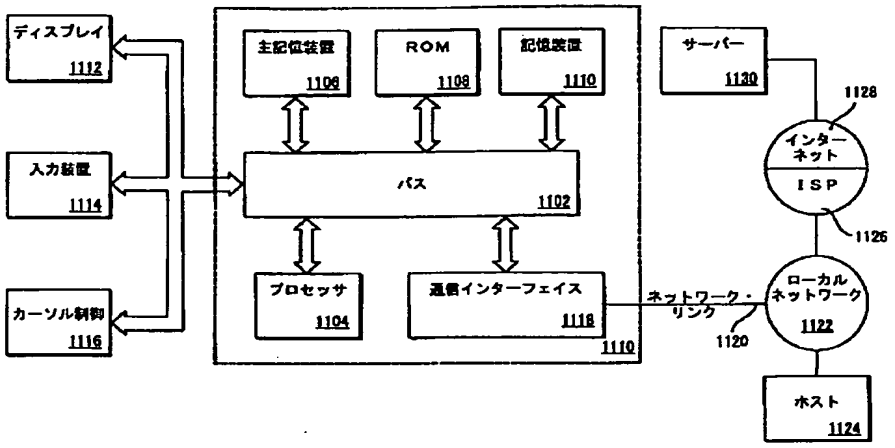
【図11】



【図13】



【図14】



フロントページの続き

(72)発明者　ジャン・ルーヘ
アメリカ合衆国　94108　カリフォルニア州　サンフランシスコ　ストックトン　ストリート　ナンバー9　540